Annals of Actuarial Science (2023), **17**, pp. 606–642
doi:10.1017/S1748499523000088

ACTUARIAL SOFTWARE

Package AdvEMDpy: Algorithmic variations of empirical mode decomposition in Python

Cole van Jaarsveldt[1]*, Matthew Ames[2], Gareth W. Peters[3] and Mike Chantler[4]

[1]School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, UK; [2]ResilientML, Melbourne, Australia; [3]Department of Statistics & Applied Probability, University of California, Santa Barbara, CA 93106, USA; and [4]School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, UK
*Corresponding author. E-mail: cv25@hw.ac.uk

(Received 27 September 2022; revised 23 February 2023; accepted 25 February 2023; first published online 5 May 2023)

Abstract

This work presents a Python EMD package named **AdvEMDpy** that is both more flexible and generalises existing empirical mode decomposition (EMD) packages in Python, R, and MATLAB. It is aimed specifically for use by the insurance and financial risk communities, for applications such as return modelling, claims modelling, and life insurance applications with a particular focus on mortality modelling. **AdvEMDpy** both expands upon the EMD options and methods available, and improves their statistical robustness and efficiency, providing a robust, usable, and reliable toolbox. Unlike many EMD packages, **AdvEMDpy** allows customisation by the user, to ensure that a broader class of linear, non-linear, and non-stationary time series analyses can be performed. The intrinsic mode functions (IMFs) extracted using EMD contain complex multi-frequency structures which warrant maximum algorithmic customisation for effective analysis. A major contribution of this package is the intensive treatment of the EMD edge effect which is the most ubiquitous problem in EMD and time series analysis. Various EMD techniques, of varying intricacy from numerous works, have been developed, refined, and, for the first time, compiled in **AdvEMDpy**. In addition to the EMD edge effect, numerous pre-processing, post-processing, detrended fluctuation analysis (localised trend estimation) techniques, stopping criteria, spline methods, discrete-time Hilbert transforms (DTHT), knot point optimisations, and other algorithmic variations have been incorporated and presented to the users of **AdvEMDpy**. This paper and the supplementary materials provide several real-world actuarial applications of this package for the user's benefit.

Keywords: Empirical Mode Decomposition (EMD); Statistical EMD (SEMD); Enhanced EMD (EEMD); Ensemble EMD; Hilbert transform; Time series analysis; Filtering; Graduation; Winsorization; Downsampling; Splines; Knot optimisation; Python; R; MATLAB

Software Availability

The software accompanying this paper is available on GitHub at:

`https://github.com/Cole-vJ/AdvEMDpy`.

Instructions on how to install, extensive worked examples, as well as the package versions required for complete reproducibility can all be found in the repository.

1. Software Contributions and Context

Increasingly with modern data analytics and machine learning entering into the domain of actuarial science through areas of consumer analytics, cyber risk modelling, sophisticated mortality models, individual claims assessment, rate making, and interest rate dynamics to name a few areas where time series analysis continues to be instrumental, we have seen the emergence of modern techniques for treating such time series analysis. This includes state space modelling, deep learning, variational autoencoders (VAE), and Gaussian processes (GP) to list a few techniques being developed. However, to date there has been little development of non-linear and non-stationary time series methods or tool boxes for actuarial settings that are based upon the highly utilised signal decomposition framework known as empirical mode decomposition (EMD). It is the intention of this work to achieve two primary objectives: the first is to introduce the basic framework of the EMD method for the actuarial audience and the second is to provide a state-of-the art Python toolbox that has been developed to facilitate easy adoption of this powerful family of non-linear and non-stationary time series methods. For more details of EMD, see van Jaarsveldt *et al.* (2021).

EMD is a non-constructive, explicit basis deconstruction, time-frequency analysis technique that is better suited to more complex non-linear and non-stationary univariate time series analysis than classical time series and time-frequency methods. The technique was initially introduced and developed in Huang *et al.* (1998, 1999) and Huang (1999). These seminal works introduced the method with minimal algorithmic variations, but numerous works over the last two decades have attempted to refine the method both formally and heuristically. Cubic splines were the original spline technique used – the mean squared error (MSE) optimality of which is shown rather elegantly in Craven & Wahba (1978), Wahba (1990), and Bowman & Evers (2017). In Chen *et al.* (2006) cubic B-splines are used, whereas in Egambaram *et al.* (2016) cubic Hermite splines and Akima splines are used. Cubic B-splines will be the focus of this work owing to the optimality of cubic splines and the iterative and closed-form solution to the Hilbert transform (HT) of cubic B-splines. Other cubic splines are possible, such as the cubic truncated power series, but this is not discussed or implemented in **AdvEMDpy** owing to the non-compactness of these bases.

The EMD package in R Core Team (2017), Kim & Oh (2018), is the current state of the art. Two separate packages in Python Core Team (2019), Laszuk (2020) and Quinn (2020), have been published more recently and are more frequently updated with the most recent updates having been published this year. The MathWorks (2021) EMD package Ortigueira (2021) is also included here to compare and contrast with the other developed EMD packages. The core advantages offered by the package developed over these existing packages will be comprehensively outlined in a detailed breakdown of the similarities and differences.

The remainder of this paper continues as follows: section 3 introduces EMD and briefly discusses the core components of the algorithm. Section 3.1 discusses the sifting mechanism designed to extract the IMFs, before the HT and resulting instantaneous frequency (IF) are discussed in section 3.2. The specific case of cubic B-spline EMD is discussed in section 3.3. In section 4, the other EMD packages available in Python, R, and MATLAB are presented. In section 4.1, the Python EMD package by Quinn (2020) is discussed as well similarities and differences with this package. In the next section, section 4.2, the other Python package by Laszuk (2020) is discussed. The two Python EMD packages are similar to each other and are in earlier stages of development than the EMD package discussed in section 4.3. This package developed by Kim & Oh (2018) in R contains more algorithmic variations to the core univariate EMD algorithm. Finally, the MATLAB package, Ortigueira (2021), is discussed in section 4.4. Section 5 summarises the proposed features of **AdvEMDpy** with the differences between **AdvEMDpy** and the other available packages being tabulated for ease of reference in Table 1. In section 6, some easily translatable pseudo-code is introduced that summarises the methodology used to implement EMD.

In the following sections, the developed package presented in this paper, **AdvEMDpy**, is discussed. In section 7, the nuances of the implementation of the algorithm are discussed in detail.

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

Table 1. Table displaying which features are present in **AdvEMDpy** package versus other available packages in R, MATLAB, and Python.

				Python:	
Features	AdvEMDpy	R: EMD	MATLAB: EMD	emd	PyEMD
Intermediate Reporting Text: section 7.1	✓	✗	✗	✗	✗
Intermediate Reporting Plots: section 7.1	✓	✓	✗	✗	✗
Pre-processing: section 7.2	✓	✗	✗	✗	✗
Symmetric Edge Effects: section 7.3	✓	✓	✓	✓	✓
Sloped-Based Edge Effects: section 7.3	✓	✗	✗	✗	✗
Characteristic Wave Edge Effects: section 7.3	✓	✓	✗	✗	✗
Explicit Edge Effects: section 7.3	✓	✗	✗	✗	✗
Statistical EMD: section 7.4	✓	✓	✗	✗	✗
Enhanced EMD: section 7.4	✓	✗	✗	✗	✗
Inflection Point Interpolation: section 7.4	✓	✗	✗	✗	✗
Binomial Average Interpolation: section 7.4	✓	✗	✗	✗	✗
Modified Mean Threshold: section 7.5	✓	✗	✓	✗	✗
Maximum IMFs: section 7.5	✓	✓	✓	✗	✗
Fixed Iteration: section 7.5	✓	✓	✓	✓	✓
S Stoppage: section 7.5	✓	✓	✗	✗	✓
Cauchy-Type Convergence: section 7.5	✓	✓	✓	✓	✓
Cauchy-Type Convergence 11a: section 7.5	✓	✓	✗	✗	✗
Cauchy-Type Convergence 11b: section 7.5	✓	✓	✗	✗	✗
Mean Fluctuation Threshold: section 7.5	✓	✓	✗	✓	✗
Energy Difference Tracking: section 7.5	✓	✗	✗	✗	✗
Cubic B-Splines: section 7.6	✓	✗	✗	✓	✗
Cubic Hermite Splines: section 7.6	✓	✗	✓	✓	✗
Akima Splines: section 7.6	✓	✗	✗	✗	✓
Basic DTHT: section 7.7	✓	✗	✗	✗	✗
FFT DTHT: section 7.7	✓	✗	✗	✗	✗
Knot Optimisation: section 7.8	✓	✗	✗	✗	✗
Ensemble EMD (EEMD): section 7.9	✓	✗	✗	✓	✓
Complete EEMD (CEEMD)	✗	✗	✗	✓	✓
Bi-dimensional (BEMD)	✗	✓	✗	✗	✓

In section 7.1, the base implementation of the algorithm is discussed as well as some helpful optional outputs such as a debugging mode plotting each iterative IMF candidate and calculated local means, text output displaying the intermediate results of stopping criterion tests, and other outputs. In section 7.2, several pre-processing techniques are discussed that can be broadly grouped into filtering and smoothing. In section 7.3, the most ubiquitous problem in EMD, the different ways of dealing with edge effects, is discussed. After discussing the edge effects, the different techniques available to isolate the local means are discussed in section 7.4. As an alternative to over-sifting as a result of restrictive IMF definitions, the other stopping criteria are discussed in section 7.5. Following the alternative stopping criteria, the other available spline methods are discussed in section 7.6. Cubic B-spline EMD allows a closed-form solution to the HT of the

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

IMFs extracted. A natural extension to the previous section, discrete-time HTs, is described in section 7.7. This is necessary for the first IMF should initial smoothing not be done or should an alternative spline fitting technique be used. In section 7.8, two knot optimisation techniques are discussed. Finally, Ensemble EMD is discussed in section 7.9. Some worked examples (synthetic and real-world) are performed in section 8 (and in the supplementary materials) to demonstrate how simple implementations can be performed alongside some figures demonstrating **AdvEMDpy**'s superior performance compared against PyEMD 0.2.10 and emd 0.3.3.

2. Actuarial Setting and Context

EMD and its algorithmic variations are important to numerous fields within actuarial science. In applying EMD to actuarial time series, one introduces a family of solutions to deal with non-stationary and non-linear structures and further decomposes these into a variety of multiple time-frequency components. This decomposition technique can be applied specifically to life expectancy modelling, mortality rate modelling, long-term forecasting, longevity curves, catastrophe modelling, cyber crime analysis, loss event analysis, crime data, health analytics data, predictive analytics data, sensor data such as in real-time insurance applications for driving and manufacturing settings, vehicle collision modelling, heavy machinery fault detection and premium adjustment, travel insurance modelling, agricultural insurance, and many others.

In Tabrizi *et al.* (2014), Wu & Qu (2008), Rezaei & Taheri (2011, 2010), and Du *et al.* (2012), EMD is used to decompose time series data from roller bearing machinery, rotating machinery, building structural components, industrial structural components, and motor vehicles, respectively, to assist in the early detection of faults. In Zheng *et al.* (2010), Liu *et al.* (2015), Wu *et al.* (2017), Hao *et al.* (2020), and Jin *et al.* (2020), EMD is used to assist in predicting agricultural yields. In Salisbury & Sun (2004), El-Kafrawy *et al.* (2014), Egambaram *et al.* (2016), Huang & Ling (2022), and Sun *et al.* (2012), EMD is used to better detect anomalies in electrocardiograms (EEC), electroencephalograms (EEG), and photoplethysmograms (PPG), which has clear applications in health and malpractice insurance. Recently, EMD has also found applications in fraud analytics for speech signals when accessing automated private databases such as voice access to banking records, see Campi *et al.* (2021).

With the formalisation of banking specialisation in actuarial science with the addition of the B100 (Banking Principles) and B200 (Banking Applications) exams, the application of EMD to financial data such as in Drakakis (2008), Guhathakurta *et al.* (2008), Lin *et al.* (2012), Wang & Wang (2017), and Nava *et al.* (2018) is used to quantify and forecast financial risk. In the papers that originally proposed EMD, Huang *et al.* (1998) and Huang *et al.* (1999), seismic data, anaemic data, and hydrological data are analysed in a better resolution than traditional techniques which have various applications in catastrophe modelling, extreme weather insurance, travel insurance, and industrial trade and commerce insurance. Numerous other works also explore these areas such as Chen *et al.* (2006) and Huang & Wu (2008) also investigating seismic data, Coughlin & Tung (2004) investigating solar cycles with applications in extreme solar flare forecasting for both personal and industrial electromechanical insurance, and Chiew *et al.* (2005) investigating hydrographic data with applications in industrial trade and commerce insurance.

EMD has been further applied in the prediction and analysis of criminal activity which with no ambiguity can be applied to general insurance and homeowners insurance. In de Freita Lucena *et al.* (2021), EMD is used to investigate numerous different criminal activities with the emphasis being on predicting and forecasting future criminal activity. In Aziz *et al.* (2020), EMD is used to investigate electrical theft. In Ahmad *et al.* (2020), EMD is used successfully to detect criminal activity using sonic data which could be used to improve security systems which are linked to lower and more competitive premiums.

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

3. Empirical Mode Decomposition

The EMD algorithm and the Hilbert-Huang transform (HHT) are often used interchangeably to refer to the same procedure in the literature. EMD primarily refers to the sifting iterative algorithmic procedure discussed in the next section, whereas the HHT refers to both the sifting procedure proposed by Huang *et al.* (1998) followed by the HT of the resulting IMFs to produce meaningful IFs discussed in section 3.2. Cubic B-spline EMD is discussed in section 3.3.

3.1 Sifting

The algorithm will be outlined here as it was introduced in Huang *et al.* (1998) before the various extensions are discussed. To this end, some notation is introduced here. Let $|\{\cdot\}|$ be the cardinality of the set, $\gamma_k(t)$ be the kth IMF defined over $t \in [0, T]$, $\tilde{\gamma}_k^M(t)$ be the maxima envelope fitted through the maxima, $\tilde{\gamma}_k^m(t)$ be the minima envelope fitted through the minima, and let $\tilde{\gamma}_k^\mu(t)$ be the mean envelope being the average of $\tilde{\gamma}_k^M(t)$ and $\tilde{\gamma}_k^m(t)$, with $\frac{d\gamma_k(t)}{dt}$ and $\frac{d^2\gamma_k(t)}{dt^2}$ being the first and second derivatives, respectively, of the IMF, $\gamma_k(t)$. With this notation, the two defining conditions of an IMF (first introduced by Campi *et al.* 2021) are expressed mathematically as below:

Condition 1 $$\text{abs}\left(\left|\left\{\frac{d\gamma_k(t)}{dt} = 0 : t \in (0, T)\right\}\right| - \left|\left\{\gamma_k(t) = 0 : t \in (0, T)\right\}\right|\right) \leq 1, \text{ and}$$

Condition 2 $$\tilde{\gamma}_k^\mu(t) = \left(\frac{\tilde{\gamma}_k^M(t) + \tilde{\gamma}_k^m(t)}{2}\right) = 0 \; \forall \; t \in [0, T] \text{ with,}$$

$$\gamma_k(t) = \tilde{\gamma}_k^M(t) \text{ if } \frac{d\gamma_k(t)}{dt} = 0 \text{ and } \frac{d^2\gamma_k(t)}{dt^2} < 0,$$

$$\gamma_k(t) \leq \tilde{\gamma}_k^M(t) \; \forall \; t \in [0, T],$$

$$\gamma_k(t) = \tilde{\gamma}_k^m(t) \text{ if } \frac{d\gamma_k(t)}{dt} = 0 \text{ and } \frac{d^2\gamma_k(t)}{dt^2} > 0, \text{ and}$$

$$\gamma_k(t) \geq \tilde{\gamma}_k^m(t) \; \forall \; t \in [0, T].$$

This definition, specifically Condition 2, while precise in its formulation of one of the characteristics of an IMF that are to be extracted by the EMD sifting procedure, is not practical for implementation. Satisfying this condition is difficult to attain in practice and may lead to an excessive computational effort which can also often lead to the propagation of errors and, potentially, the removal of meaningful structures from the signal basis representation. As such, one often needs to apply an approximation that is more computationally efficient and iteratively favourable. Such a definition can be seen below:

Modified Condition 2 $$\sum_t \left|\tilde{\gamma}_k^\mu(t)\right| \leq \epsilon,$$

for some chosen ϵ with $\tilde{\gamma}_k^\mu(t)$ defined discretely ($t \in \{t_0, \ldots, t_N\}$) during implementation. If Condition 1 and Modified Condition 2 are met, then the IMF is stored and removed from the remainder or residual ($r_k(t) = r_{k-1}(t) - \gamma_k(t)$), if either are not met then the process is repeated (or an additional optional third condition, known as a stopping criterion, is checked). The process is repeated until an IMF candidate either meets the required conditions or a stopping criterion is met. The end result of the sifting procedure will be an IMF representation of the original time series, $x(t)$, with the total number of IMFs being denoted by K and the remainder or trend being denoted by $r_K(t)$:

$$x_{IMF}(t) = \sum_{k=1}^{K} \gamma_k(t) + r_K(t). \tag{1}$$

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

3.2 Hilbert transform and instantaneous frequency

Once the basis of the time series has been extracted, they need to be analysed using the HT to provide a time-frequency characterisation. The IMF basis represents the time-domain signal decomposition, and the objective is to transform this basis decomposition into a frequency-based characterisation. The joint process of decomposing the time series using EMD and Hilbert transforming the result is known as the HHT. The HT of a time series only makes physical sense if the time series satisfies the IMF conditions. Given an IMF, $\gamma_k(t)$, the corresponding HT, $\check{\gamma}_k(t) = \mathcal{HT}[\gamma_k(t)]$ with $\mathcal{HT}[\cdot]$ being the HT, is calculated as follows:

$$\check{\gamma}_k(t) = \frac{1}{\pi} PV \int_{-\infty}^{\infty} \frac{\gamma_k(t^*)}{t - t^*} dt^*, \tag{2}$$

with PV being the Cauchy Principle Value integral. It is an improper integral that assigns values to otherwise undefined integrals. The Cauchy principal value integral is formally defined as:

$$\check{\gamma}_k(t) = \frac{1}{\pi} \lim_{\epsilon \to 0^+} \left[\int_{t-\frac{1}{\epsilon}}^{t-\epsilon} \frac{\gamma_k(t^*)}{t - t^*} dt^* + \int_{t^*+\epsilon}^{t+\frac{1}{\epsilon}} \frac{\gamma_k(t^*)}{t - t^*} dt^* \right]. \tag{3}$$

The analytical signal, $\gamma_k^a(t)$, can then be defined as:

$$\gamma_k^a(t) = \gamma_k(t) + i\check{\gamma}_k(t) = a_k(t) e^{i\theta_k(t)} = a_k(t) e^{i \int \omega_k(t) dt}, \tag{4}$$

with $a_k(t)$ and $\theta_k(t)$ being defined as:

$$a_k(t) = \sqrt{\gamma_k(t)^2 + \check{\gamma}_k(t)^2}, \tag{5}$$

and

$$\theta_k(t) = \tan^{-1}\left(\frac{\check{\gamma}_k(t)}{\gamma_k(t)}\right). \tag{6}$$

The IF, $\omega_k(t)$, for the k-th IMF, is then defined as:

$$\omega_k(t) = \frac{d\theta_k(t)}{dt}. \tag{7}$$

The Hilbert spectrum of IMF $\gamma_k(t)$ is calculated as:

$$H_k(t) := H(t; \omega_k) = \begin{cases} a_k(t), & \omega_k \le \omega(t) < \omega_{k+1} \\ 0, & \text{otherwise} \end{cases}, \tag{8}$$

with ω_k and ω_{k+1} being the discretisation of the frequency domain for analysis and plotting purposes – higher resolutions require higher computational expense and without any smoothing (Gaussian smoothing recommended – see Figures 13 and 14 for context) would be imperceptible to most in plots. The Hilbert spectrum allows for a far higher resolution than other frequency analysis techniques such as the short-time Fourier transform and Morlet wavelet transform.

3.3 Cubic B-spline empirical mode decomposition

One will immediately notice that the conditions on an IMF are characterising conditions but they are not constructive; in other words, one must still define a functional form for parameterisation of the representation of the IMFs. This is distinct from methods such as Fourier decomposition where a fixed basis, in this case, cosines, is determined as fundamentally part of the basis decomposition. This is both advantageous and challenging for EMD methods. On the one hand, it provides a lot of flexibility to the modeller to achieve the conditions for the IMF in the EMD sifting procedure; on the other hand, it opens up a model specification and selection problem. Fortunately, to

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

some extent, this question has been addressed in the literature, at least to the extent that numerous choices have been proposed.

This section follows the work done in Chen *et al.* (2006) on applying cubic B-splines to EMD. In this case, one expresses the EMD representation as follows:

$$x_{IMF}(t) = \sum_{k=1}^{K} \gamma_k(t) + r_K(t), \tag{9}$$

with each $\gamma_k(t)$ and the trend, $r_K(t)$, being the sum of compact cubic B-splines. The optimality of cubic splines concerning MSE minimisation has been shown in Craven & Wahba (1978), Wahba (1990), and Bowman & Evers (2017), and in addition, the cubic B-splines have the further practical advantage that the HT of B-splines can be obtained in closed-form in a recursive representation. B-splines and the recursive relationship that exists to define the higher-order splines are shown in de Boor (1978). The B-spline of order 1 is shown below:

$$B_{i,1,\boldsymbol{\tau}}(t) = \mathbb{1}_{[\tau_i, \tau_{i+1})}. \tag{10}$$

This is the base case and as such the higher-order splines can be defined recursively. $B_{i,j,\boldsymbol{\tau}}(t)$ is the ith basis function of order j defined over knot sequence $\boldsymbol{\tau}$ with $t \in \mathbb{R}$. With this framework, $B_{i,j,\boldsymbol{\tau}}(t)$ is defined recursively as:

$$B_{i,j,\boldsymbol{\tau}}(t) = \frac{t - \tau_i}{\tau_{i+j-1} - \tau_i} B_{i,j-1,\boldsymbol{\tau}}(t) + \frac{\tau_{i+j} - t}{\tau_{i+j} - \tau_{i+1}} B_{i+1,j-1,\boldsymbol{\tau}}(t). \tag{11}$$

Equation (11) is proven in de Boor (1978). The HT of Equation (10), with $\check{B}_{i,1,\boldsymbol{\tau}}(t) = \mathcal{HT}[B_{i,1,\boldsymbol{\tau}}(t)]$, can be shown to be:

$$\check{B}_{i,1,\boldsymbol{\tau}}(t) = \frac{1}{\pi} ln \left| \frac{t - \tau_i}{t - \tau_{i+1}} \right|, \tag{12}$$

with singularities at τ_i and τ_{i+1}. It is for this reason that the knot sequence is advised to be a proper subset of the time series. The corresponding HT recursive relationship is proven in Chen *et al.* (2006) and is stated below:

$$\check{B}_{i,j,\boldsymbol{\tau}}(t) = \frac{t - \tau_i}{\tau_{i+j-1} - \tau_i} \check{B}_{i,j-1,\boldsymbol{\tau}}(t) + \frac{\tau_{i+j} - t}{\tau_{i+j} - \tau_{i+1}} \check{B}_{i+1,j-1,\boldsymbol{\tau}}(t). \tag{13}$$

With this framework in place, the cubic B-spline fitting problem can be stated. The time series is defined at discrete points, $\mathbf{t} = \{t_0, \ldots, t_N\}$, so that the time series can be represented by a vector, $\boldsymbol{s}$, below:

$$\boldsymbol{s} = \begin{bmatrix} s(t_0) \\ s(t_1) \\ \vdots \\ s(t_N) \end{bmatrix}. \tag{14}$$

The knot sequence is implied by expressing the problem objective function in matrix form, and as such, the knot sequence subscript is dropped below:

$$\boldsymbol{B} = \begin{bmatrix} B_{0,4}(t_0) & \cdots & B_{(M-4),4}(t_0) \\ B_{0,4}(t_1) & \cdots & B_{(M-4),4}(t_1) \\ \vdots & \ddots & \vdots \\ B_{0,4}(t_N) & \cdots & B_{(M-4),4}(t_N) \end{bmatrix}. \tag{15}$$

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

Finally the coefficient vector, $\boldsymbol{c}$, is defined below:

$$\boldsymbol{c} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{(M-4)} \end{bmatrix}. \tag{16}$$

Using the above, the objective function can be defined as follows:

$$MSE(\boldsymbol{c}|\boldsymbol{s}) = (\boldsymbol{s} - \boldsymbol{B}\boldsymbol{c})^T(\boldsymbol{s} - \boldsymbol{B}\boldsymbol{c}). \tag{17}$$

The result of this optimisation is a cubic B-spline interpolation spline, $p(t)$, such that:

$$p(t) = \sum_{i=0}^{M-4} c_i B_{i,4,\boldsymbol{\tau}}(t). \tag{18}$$

The power of using cubic B-splines is that the HT of this spline, $\check{p}(t) = \mathcal{HT}[p(t)]$, is simply:

$$\check{p}(t) = \sum_{i=0}^{M-4} c_i \check{B}_{i,4,\boldsymbol{\tau}}(t). \tag{19}$$

Several variations of this general framework will be explored. An explicit smoothing term can be incorporated in Equation (17) to create explicitly smoothed cubic B-splines as in Equation (36). These discretely penalised B-splines are referred to as P-splines. The number and distribution of knots can be changed to implicitly smooth the spline either manually or using knot point optimisation as in section 7.8, and the knot points are extended beyond the edge of the time series to create a non-natural spline to better fit the edges of the time series as shown in Figure 2.

Assuming a consistent set of knots throughout the sifting procedure (either `optimise_knots=0` or `optimise_knots=1`, but more about this in sections 7.1 and 7.8) and by letting remaining trend structure K, $r_K(t)$, be denoted as structure $K+1$, Equation (9) can be expressed as:

$$\begin{aligned} x_{IMF}(t) &= \sum_{k=1}^{K} \sum_{i=0}^{M-4} c_{i,k} B_{i,4,\boldsymbol{\tau}}(t) + r_K(t) \\ &= \sum_{k=1}^{K+1} \sum_{i=0}^{M-4} c_{i,k} B_{i,4,\boldsymbol{\tau}}(t), \end{aligned} \tag{20}$$

with $c_{i,k}$ denoting coefficient i for structure k.

4. Existing Available EMD Packages

Before introducing the available algorithmic variations of EMD in this package, one would benefit from a summary of the other available EMD packages as well as the other available algorithmic variations. This package does not seek to address multi-dimensional EMD as it deserves significant research and dedication in its own right. The other packages available are in different stages of development – the two Python packages and the MATLAB package are more frequently updated and are therefore less complete than the R package. While, being more complete, the options available, particularly concerning the edge effect, local mean estimation, pre-processing, stopping criteria, and knot point optimisation, are still lacking which **AdvEMDpy** seeks to address.

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

This section, as well as a comparison with the available features in **AdvEMDpy**, is summarised in Table 1.

4.1 Python – emd 0.3.3

In this package, Quinn (2020), there are numerous variations of EMD available. The standard univariate EMD does not have many algorithmic variations. The stopping criteria used are the Cauchy-type convergence (section 10.5.4 of the supplement), mean fluctuation threshold (section 10.5.7 of the supplement), and fixed iteration (section 10.5.2 of the supplement). The splines used in the packages are cubic B-splines, piece-wise cubic Hermite splines, and monotonic piece-wise cubic Hermite splines. The edges are reflected to prevent extreme extrapolation at the edges causing errors to propagate. Ensemble EMD (EEMD) (Wu & Huang, 2009) (section 7.9), complete EEMD (CEEMD) (Torres *et al.*, 2011), masking signal-assisted EMD (Deering & Kaiser, 2005), and Holo-Hilbert spectral analysis (Huang *et al.*, 2016) are also available. CEEMD adds noise to every IMF candidate, rather than once at the start of each iteration of the algorithm – this does show promising results but deserves further study with an appropriate sorting mechanism. Adding a masking signal is very time series specific – it depends on the frequency and amplitude composition of the IMF candidates and therefore usually requires an initial unmasked sifting. Holo-Hilbert spectral analysis attempts to handle more complex time series combinations.

4.2 Python – PyEMD (EMD-signal) 0.2.10

In this package, Laszuk (2020), some variations of EMD are available. Univariate EMD does have limited algorithmic variations. The splines available are cubic, Akima (section 10.6.2 of the supplement), and linear. The stopping criteria used are the standard Cauchy-type convergence (section 10.5.4 of the supplement), fixed iteration (section 10.5.2 of the supplement), and S Stoppage (section 10.5.3 of the supplement). Extrema are calculated using standard finite difference methods or by fitting a parabola and inferring extrema from the parabola. Extrema at the edges are reflected to deal with the edge effect. Different algorithm variations are also available: ensemble EMD (EEMD) (Wu & Huang, 2009) (section 7.9), complete EEMD (CEEMD) (Torres *et al.*, 2011), and two-dimensional EMD grouped into Bi-dimensional EMD (BEMD) and two-dimensional EMD (EMD2D) – both have been classified as in-progress or abandoned projects.

4.3 R – EMD 1.5.8

This package, Kim & Oh (2018), is a more complete and refined EMD library. The stopping criteria used are fixed iteration (section 10.5.2 of the supplement), S Stoppage (section 10.5.3 of the supplement), modified mean (section 10.5.1 of the supplement), Cauchy-type converge (section 10.5.4 of the supplement), Cauchy-type 11a (section 10.5.5 of the supplement), and Cauchy-type 11b (section 10.5.6 of the supplement). The optional edge effects are one of the symmetric edge effects discussed in section 7.3.1, the anti-symmetric edge effect (section 7.3.1), the Huang characteristic wave (section 10.3.3 of the supplement), and the Coughlin characteristic wave (section 10.3.3 of the supplement). Cubic spline interpolation is done. Furthermore, several smoothing techniques are used such as spline smoothing, kernel smoothing, and local polynomial smoothing. As a result, various forms of statistical EMD (SEMD) are available. Plotting of individual IMF candidates and final IMFs is available for debugging purposes, and certain periods may be excluded in a different method to prevent mode mixing. Confidence limits for the IMFs may also be calculated using vector auto-regressive techniques. Bi-dimensional EMD is available as well as a de-noising version of EMD. A limit can be imposed on the maximum number of IMFs.

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

4.4 MATLAB – Empirical Mode Decomposition 1.0.0.0

The package, Ortigueira (2021), is older and more tested with few, if any, recent additions. Cauchy-type convergence (section 10.5.4 of the supplement) is the stopping criterion used in this package. Fixed iterations (section 10.5.2 of the supplement) is another criterion used. An energy ratio stopping criterion is used that may be directly translated to a rephrasing of a relative modified mean criterion (section 10.5.1 of the supplement). A limit is imposed on the allowed number of IMFs. Spline methods used are cubic splines and piece-wise cubic Hermite splines. This package, while having few algorithmic variations or extensions, is well-tested – much like Kim & Oh (2018).

5. Proposed Features of AdvEMDpy Package

In this package, the univariate EMD features from the available packages will be expanded upon, and additional features that form core components of the sifting algorithm, and as such deserve more content, will be introduced and explained. In section 7.2 several pre-processing methods, a feature not introduced or mentioned in the available packages, such as mean filters, median filters, Winsorisation, Winsorisation Interpolation, initial smoothing, a Generalised Hodrick-Prescott filter, a Henderson-Whittaker filter, and downsampling, will be explained.

In section 7.3, various families of edge effect techniques are introduced that, while most are introduced in the available literature, are given a very brief treatment in the other packages. The symmetric family of methods are introduced that includes the symmetric discard method, the symmetric anchor method, the conditional symmetric anchor method, the symmetric method, and the anti-symmetric method. The family of slope-based methods includes the slope-based method and the improved slope-based method. The characteristic wave family included the modified Huang characteristic wave, the Coughlin characteristic wave, and the average characteristic wave. Finally, the first of the family of explicit methods is introduced that explicitly extrapolates the time series to find the extrema and is named the single neuron neural network. Of all these methods, only the symmetric method, anti-symmetric method, Huang characteristic wave, and Coughlin characteristic wave are used in the currently available packages.

In all available packages, only the envelope local mean estimation technique or detrended fluctuation analysis technique is available. In section 7.4, the envelope local estimation technique is discussed as well as the explicitly smoothed version referred to as the statistical EMD or SEMD using discretely penalised B-splines known as P-splines. In addition to these, enhanced EMD is used where the optimal knot points are calculated, and these are kept constant until the IMF is extracted whereupon they are recalculated for the next iteration. Inflection point interpolation where the local mean is estimated using the inflection points rather than creating and averaging envelopes and the binomial average interpolation where the local average of the time series is calculated using the binomial average of the surrounding time points. These all add to a far more robust repertoire of detrended fluctuation techniques.

Many, but not all, of the stopping criteria introduced in this package feature in other packages. The modified mean threshold is a very notable addition in this package and detailed in section 7.5 and should be used, where possible, to prevent significant over-sifting of the time series. The fixed iteration stopping criterion, S Stoppage criterion, the various Cauchy-type convergence criteria, and the mean fluctuation threshold do feature in other packages, but they are all given a thorough and meaningful treatment in this paper. The energy difference tracking criterion is a notable addition that takes advantage of the local orthogonality of the IMF bases where the sifting stops when the energy difference falls below a certain threshold.

In section 7.6, cubic Hermite spline interpolation (CHSI) is introduced. This method features in most of the other packages, but the modified version of CHSI known as Akima splines is also introduced. This spline technique in addition to the base implementation of the cubic B-splines is

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

unique to this package. The basic DTHT and the FFT DTHT that takes advantage of the relationship between the Fourier transform and the HT are both introduced in section 7.7 as alternatives to the cubic Hilbert B-spline interpolation that is also available. All these HTs are unique to this package.

This package is also the first to explicitly incorporate knot optimisation into the EMD algorithm. It is unfortunately lacking in other packages despite the clear advantage of optimal placement of the knot points. The bisection method, as well as the serial bisection method, are both introduced in section 7.8. Finally, ensemble EMD (EEMD) is introduced in section 7.9. This is available in other packages. The numerous extensions to known methods in this package as well as the formal extensions to previously neglected stages of the algorithm deserve a more ubiquitous presence in EMD and related trend analysis methods.

6. Pseudo-Code for Core Sifting Routine in AdvEMDpy Package

Before introducing Algorithm 1, one needs some notation. Let $x(t)$ be the time series defined over $\mathbf{t} = \{t_0, \ldots, t_N\}$, $h_{(p,q)}(t)$ be the qth potential iteration of IMF p, $r_p(t)$ be the pth residual, $M(t_i)$ be the maxima defined over $\mathbf{t_M} \subset \mathbf{t}$, $m(t_j)$ be the minima defined over $\mathbf{t_m} \subset \mathbf{t}$, $\tilde{h}^M(t)$ be the maxima envelope, $\tilde{h}^m(t)$ be the minima envelope, $\tilde{h}^\mu(t)$ be the mean envelope, and $\gamma_p(t)$ be IMF p. In addition to Condition 1 and Modified Condition 2, to prevent over-sifting one can impose a third condition known as a stopping criterion which is discussed in section 7.5. With an example of a stopping criterion defined in Condition 3 below after Condition 1 and Modified Condition 2 are restated for ease of reference:

Condition 1 $$\text{abs}\left(\left|\left\{\frac{d\gamma_k(t)}{dt} = 0 : t \in (0, T)\right\}\right| - \left|\left\{\gamma_k(t) = 0 : t \in (0, T)\right\}\right|\right) \leq 1,$$

Modified Condition 2 $$\sum_t \left|\tilde{\gamma}_k^\mu(t)\right| \leq \epsilon_2, \text{ and}$$

Condition 3 $$SD_{(p,q)} = \sum_{t=t_0}^{t_N} \left[\frac{\left|(h_{(p,q-1)}(t) - h_{(p,q)}(t))\right|^2}{h_{(p,q-1)}^2(t)}\right] < \epsilon_3,$$

for some ϵ_2 and ϵ_3 with ϵ_3 explained in section 10.5.4 of the supplement. The EMD sifting algorithm that extracts IMFs in an iterative envelope residual compensation method is outlined in Algorithm 1.

The first iteration of the iterative sifting algorithm outlined in Algorithm 1 is demonstrated in Figure 1 for clarity. In Figure 1, $h_{(1,0)}(t)$ is the first potential candidate for IMF 1, $\tilde{h}_{(1,0)}^M(t)$ is a spline through the maxima, $\tilde{h}_{(1,0)}^m(t)$ is a spline through the minima, $\tilde{h}_{(1,0)}^\mu(t)$ is the average of the maxima and minima splines, $M(t_i)$ is the maxima of $h_{(1,0)}(t)$, and $m(t_j)$ is the minima of $h_{(1,0)}(t)$.

7. Core Details of AdvEMDpy Package

In this section, the details of each of the components of **the AdvEMDpy** package are outlined both in how to interact with the package through specific functionalities and what expectations one should have on outputs created and features that can be customised. In some sections, the algorithmic options available are discussed without giving sufficient details (owing to length constraints) with the complete text (some of which is repeated verbatim from this section for readability) being found in "Supplement to: Package **AdvEMDpy**: Algorithmic Variations of Empirical Mode Decomposition in Python".

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

Algorithm 1. Sifting Process

Require: $x(t) = h_{(1,0)}(t) = r_0(t)$
  Initialise:
  (1) $M(t_i)$ of $x(t)$
  (2) $m(t_j)$ of $x(t)$
  While $|\{M(t_i)\}| + |\{m(t_j)\}| > 2$ **do**
    fit $\tilde{h}^M(t)$ through $M(t_i)$
    fit $\tilde{h}^m(t)$ through $m(t_j)$
    calculate $\tilde{h}^\mu(t) = \dfrac{\tilde{h}^M(t) + \tilde{h}^m(t)}{2}$
    if Condition 1 and Modified Condition 2 or Condition 3 **then**
      store $h_{(p,q)}(t) = \gamma_p(t)$
      calculate $r_p(t) = r_{p-1}(t) - \gamma_p(t)$
      find $M(t_i)$ of $r_p(t)$
      find $m(t_j)$ of $r_p(t)$
    else
      calculate $h_{(p,q+1)}(t) = r_p(t) - \tilde{h}^\mu_{(p,q)}(t)$
      find $M(t_i)$ of $h_{(p,q+1)}(t)$
      find $m(t_j)$ of $h_{(p,q+1)}(t)$
    end if
  end while
  store $r_K(t)$

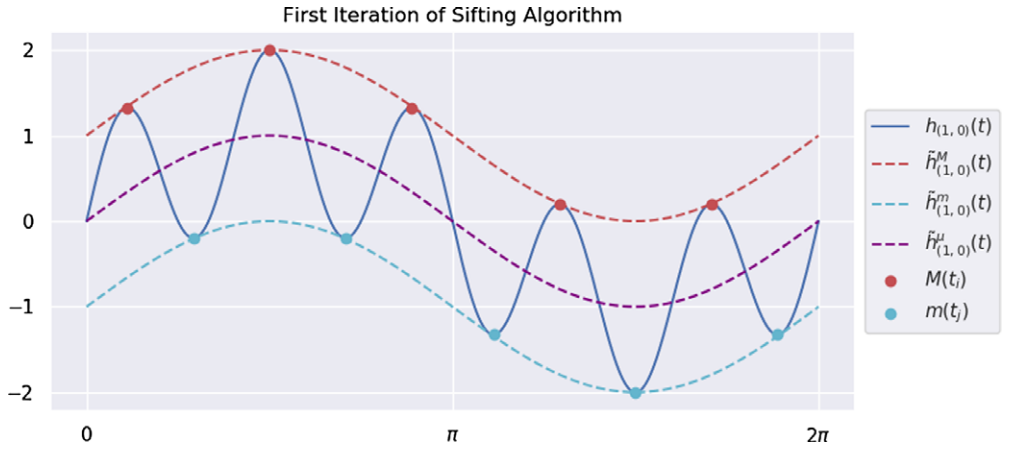

Figure 1. Figure demonstrating first iteration of Algorithm 1 where extrema are found, splines are fitted through maxima and minima, respectively, to form an envelope, and the local mean is estimated as the average of the extrema splines.

7.1 Base implementation of AdvEMDpy package

Before discussing the nuances of the algorithmic variations, one would benefit from some choice remarks concerning non-essential, but helpful outputs of the **AdvEMDpy** package. These play no direct part in the implementation of the algorithm but assist in iterative understanding and assessment of EMD's performance. The base implementation of the algorithm may be seen below. More detailed descriptions may be found in the associated scripts as well as a base implementation in the associated `README.md` file.

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

```
# instantiate EMD class with time (optional keyword argument) and time
series
emd = EMD(time=time, time_series=time_series)

# base implementation with optional helpful inputs shown
# knots and knot_time are optional keyword arguments
imfs, hts, ifs = \
    emd.empirical_mode_decomposition(knots=knots,
                                     knot_time=knot_time,
                                     debug=False,
                                     dft='envelopes',
                                     verbose=True,
                                     stopping_criterion='sd',
                                     output_coefficients=False,
                                     output_knots=False)[:3]
```

The details of each field in the input to the function are detailed in the following subsections. If time is not specified as a keyword argument, then the following code is implemented where time is simply an index set of the same length as the original time series.

```
self.time = np.arange(len(self.time_series))
```

If the knot sequence or the associated knot time are also unspecified keyword arguments, then the following code is implemented that sets knot time equal to time and spaces the knots 10 times further apart than the time points. This is problematic if the default knot sequence is insufficient to capture the highest frequency structures which is demonstrated in `new_user.ipynb`.

```
knots = np.linspace(0, self.time[-1], int(len(self.time)/10 + 1))
knot_time = self.time
```

The only required argument is the time series. This is detailed further in `new_user.ipynb` which is intended for new users to get comfortable with the package and the available options.

7.1.1 Debug flag in AdvEMDpy package

If true, each iteration of the local mean estimation through the chosen detrended mean threshold technique will be plotted for analysis. The debugging output displayed depends on the choice of `dft`. If `dft="envelopes"` (base implementation), the extrema, the extrema envelopes, and the calculated local mean will be plotted. If `dft="inflection_points"` the extrema, inflection points, and the associated spline fitted through the inflection points will be displayed. If `dft="binomial_average"` the extrema, the points used for the binomial average, the binomial average of these points, and the associated spline fitted will be displayed. Finally, if `dft="enhanced"`, the extrema, the optimal 'extrema', the associated splines, and the resulting local mean are displayed.

7.1.2 Verbose flag in AdvEMDpy package

If true, each iteration of the algorithm will output text describing if the stopping criterion chosen or the mean threshold is met. The intermediate numbering of the potential IMFs is output for consistency, and the iteration counter is printed if the maximum number of iterations is met. Different text is displayed for each of the optional stopping criteria. If, for example, `stopping_criterion='sd'` then the text displayed for the fifth

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

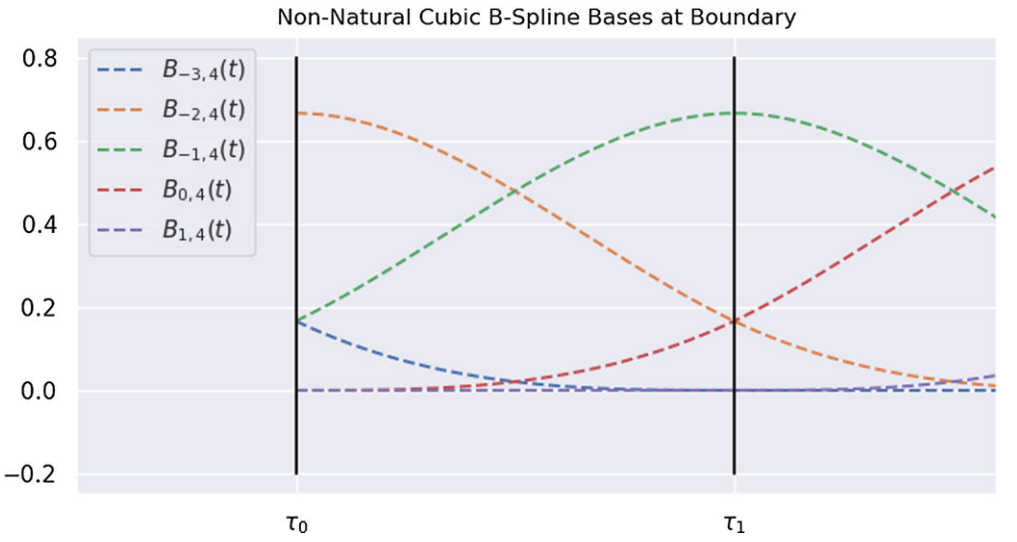

Figure 2. Figure demonstrating incomplete bases at the boundary of the time series to create non-natural cubic B-splines that can accommodate non-zero edge values, derivatives, and curvatures.

candidate for the second IMF, should the IMF candidate not meet the stopping criterion, will be `IMF_25 Standard deviation STOPPING CRITERION NOT MET with sd = 1000.0` or it will be `IMF_25 Standard deviation STOPPING CRITERION MET with sd = 0.05 < sd threshold = 0.1` with the stopping criterion threshold being another possible input such as `stopping_criterion_threshold=0.1`, if the stopping criterion condition is met. For `stopping_criterion_threshold`, a value of $0.2 - 0.3$ is recommended in the paper that originally proposed the method, Huang *et al.* (1998). This value should be time series-dependent as the value is very much dependent on the level of noise in the time series as well as the number of time points available, but more on this (and other stopping criteria) in section 7.5.

7.1.3 Output coefficients flag in AdvEMDpy package

If `output_coefficients=True`, cubic B-spline coefficients corresponding to each IMF output (including initial smoothed time series and trend) are output. One must remember the six additional coefficients corresponding to the three additional knots on either edge of the time series as demonstrated in Figure 2. These additional bases result in implicit smoothing through a non-natural spline which does not tend to zero at the edges in both position and the higher-order derivatives. It would be unnecessarily restrictive to impose a natural spline on the sifting algorithm and the resulting IMFs.

7.1.4 Output knots flag in AdvEMDpy package

If `output_knots=True`, knot points are output. If knot points are not optimised, that is if `optimise_knots=0`, then the original knot points are repeated at each iteration of the sifting procedure when extracting recursively each IMF. In other words, all IMFs will have the same knot sequence. If the user provides a uniform knot sequence and chooses not to optimise the knot locations, then a uniform knot sequence will be used throughout as demonstrated in Figure 3.

Alternatively, one can optimise the knots on an original spline (`optimise_knots=1`) representation of the input time series signal using optimal knot placement methodology as outlined in section 7.8 where two options are available using simple bisection and serial bisection, but more

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

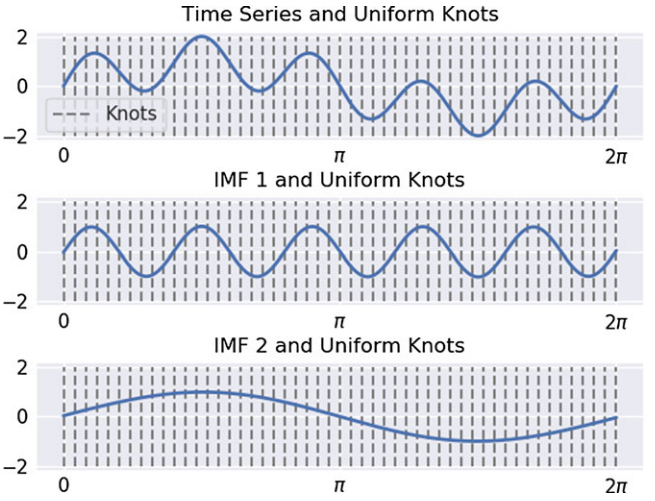

Figure 3. Figure demonstrating predefined uniform knot placement and the resulting IMFs.

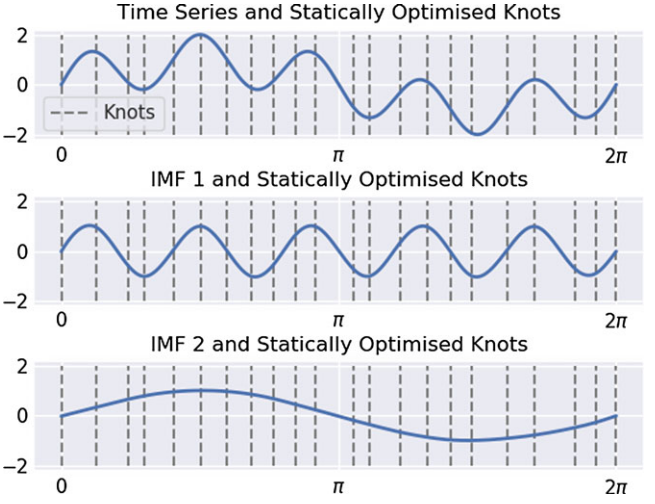

Figure 4. Figure demonstrating statically optimised knot placement, which is optimised once at outset and used throughout the sifting, and the resulting IMFs.

about this will follow in the dedicated section. In this way, the knot points will still be universal for all IMFs; however, they will be optimised for the given signal input and will not dynamically adjust as the residuals become increasingly less complex. For this reason, this can be referred to as the statically optimised knots, and an example demonstrating this can be seen in Figure 4 where the optimised non-uniform knots can be seen to be used throughout the algorithm despite the decreasing complexity.

In addition, one could also optimise the knot points per iteration (`optimise_knots=2`) of the EMD sifting procedure; once an IMF is isolated, the next iteration of EMD on the residual signal could first have the knot point optimisation performed when fitting the spline to the residual to proceed with the next rounds of sifting to extract the next IMF. This results in different numbers of knot points per IMF and different placements; in general, one would expect to require far fewer

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

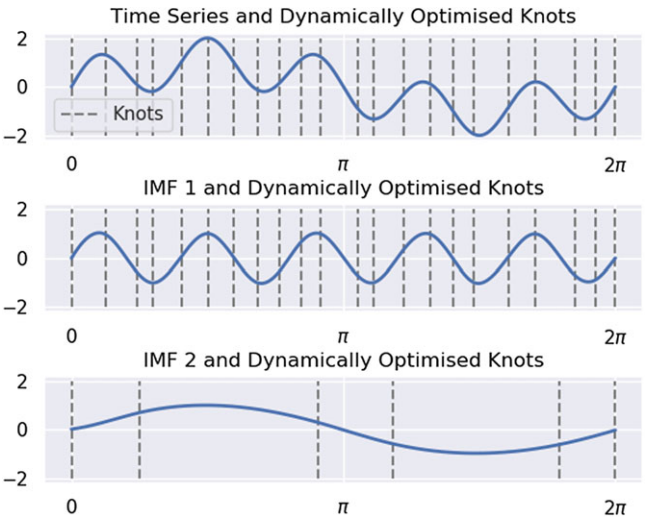

Figure 5. Figure demonstrating dynamically optimised knot placement, which is optimised at the beginning of each internal sifting routine, and the resulting IMFs.

knots for later IMFs extracted which have lower frequency content compared to those first few IMF bases extracted. An example demonstrating this can be seen in Figure 5 where there are far fewer non-uniform knots for the second IMF than was required for the first IMF.

Performing the third option may increase the speed of the algorithm if a large number of siftings need to take place and a large number of IMFs need to be extracted, despite the added time required to find the optimised knots at each stage. Progressively fewer knots will be required throughout the algorithm with the added advantage of having a more parsimonious representation with fewer parameters required for higher-order IMFs which have guaranteed reducing oscillation in their representation.

7.1.5 Recommendations for base implementation of AdvEMDpy package

For the initial application of the algorithm to the desired time series, it is recommended that the user display the results of the selected stopping criterion (`verbose=True`) and run to completion. The user can then inspect the outputs and should anything be irregular or if the user wants to observe each iterative step of the algorithm one can use `debug=True` which will cause the method to plot each iteration of the algorithm.

```
# recommended initial base implementation
imfs, hts, ifs = emd.empirical_mode_decomposition(knots=knots,
                                                  knot_time=knot_time,
                                                  debug=True,
                                                  verbose=True)[:3]
```

7.2 Pre-processing flag in AdvEMDpy package

The raw time series signals that can be passed to the EMD package may contain a wide variety of structures. This could include different non-linear structures, non-stationarity features in trend, volatility, etc., as well as corruption by observation noise that may be present in the collection of the data. Therefore, in the **AdvEMDpy** package there is the option to pre-process the time series

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

data input to reduce the effect of these features if they may be adverse to the user's analysis. This is particularly the case in the context of signals observed in various noisy environments.

Therefore, before the sifting algorithm is implemented, there are several choices available for the pre-processing of the raw time series. This was a necessary inclusion in the algorithm owing to the wide variety of time series and their potentially vastly different statistical characteristics that would need to be accommodated in the EMD basis decomposition. These pre-processing techniques may be broadly grouped into filtering and smoothing. The filtering methods developed out of a necessity to mitigate the corruption of the IMFs that would otherwise arise as a result of the permeation of error that would occur due to the presence in the input time series signal of noise corruptions such as heavy-tailed noise, mixed noise (Gaussian noise with different variances), and Poisson noise. Smoothing developed out of the broader field of trend extraction among cyclical components where within EMD defining an individual cyclical component among others and within a non-stationary setting becomes increasingly challenging. The base implementation is as follows:

```
imfs, hts, ifs = \
    emd.empirical_mode_decomposition(knots=knots, knot_time=knot_time,
                                     initial_smoothing=True,
                                     preprocess='none',
                                     preprocess_window_length=51,
                                     preprocess_quantile=0.9,
                                     preprocess_penalty=1,
                                     preprocess_order=13,
                                     preprocess_norm_1=2,
                                     preprocess_norm_2=1,
                                     downsample_window='hamming',
                                     downsample_decimation_factor=20,
                                     downsample_window_factor=20)[:3].
```

The details of the implementation of filtering techniques are available such as a mean filter, median filter, Winsorization, and Winsorization and interpolation; one can see section 10.2.2 in "Supplement to: Package **AdvEMDpy**: Algorithmic Variations of Empirical Mode Decomposition in Python". The smoothing techniques available such as generalised Hodrick-Prescott smoothing, Henderson-Whittaker graduation, downsampling, and downsampling followed by decimation can be found in section 10.2.3 of the supplementary materials.

7.2.1 Recommendations for pre-processing flag in AdvEMDpy package

The pre-processing required depends upon the level of noise present in the time series and is very time series and process dependent. There is no pre-processing to suit all time series, but the most robust version of pre-processing would be median filtering the noise out of the time series (`preprocess="median_filter"` with an appropriate window length. The window length is dependent upon the highest frequency present in the time series and the sampling rate. Following the median filter, the user should then perform an initial smoothing (`initial_smoothing=True`) to remove the discontinuities introduced by the median filter.

```
imfs, hts, ifs = \
    emd.empirical_mode_decomposition(knots=knots, knot_time=knot_time,
                                     initial_smoothing=True,
                                     preprocess='median_filter',
                                     preprocess_window_length=51)[:3]
```

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

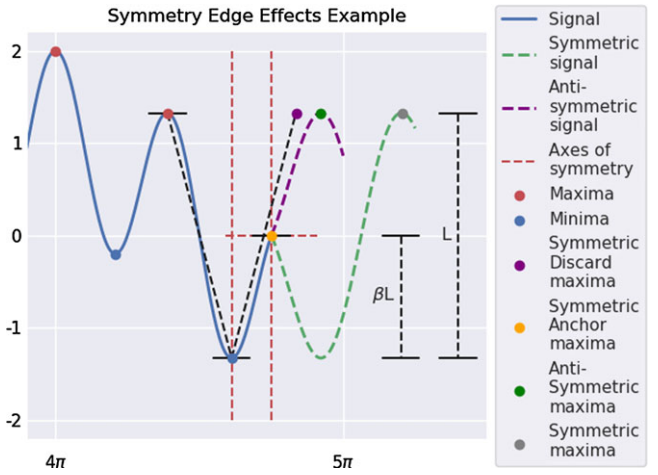

Figure 6. Example time series demonstrating four (five if conditional symmetric anchor is included) different symmetric edge effect techniques with axes of symmetry included.

7.3 Edge effects in AdvEMDpy package

The most prevalent challenge in EMD is the propagation of errors throughout the IMFs and the permeation of the errors throughout the sifting process. Owing to the iterative nature of the algorithm, incorrectly estimated edges lead to errors being ubiquitous throughout all the IMFs. There are several techniques used to estimate the extrema beyond the edges of the signal – this **AdvEMDpy** package attempts to give the reader as many reasonable and researched options as possible. The base implementation of the edge effects code is as follows:

```
imfs, hts, ifs = \
    emd.empirical_mode_decomposition(knots=knots,
                                     knot_time=knot_time,
                                     edge_effect='symmetric',
                                     sym_alpha=0.1,
                                     nn_m=200,
                                     nn_k=100,
                                     nn_method='grad_descent',
                                     nn_learning_rate=0.01,
                                     nn_iter=100)[:3].
```

7.3.1 Symmetric methods

Without loss of generality, the procedure will be explained for the right edge maxima of the signal. Figure 6 demonstrates examples of the three possible symmetric edge effects. In no particular order, the technique demonstrated in Rilling *et al.* (2003) and Wu & Qu (2008) is referred to in this paper as the symmetric discard technique owing to the discarding of the end of the time series in the new extrema approximation.

Symmetric Discard – In Figure 6, an implementation of the Symmetric Discard technique (`edge_effect="symmetric_discard"`) maximum is shown with a purple dot. The end of the time series between the last blue dot and the orange dot is disregarded when approximating the next extreme. Taking the last maximum value, $X(t_M^{\max})$, the associated time point, $t_M^{\max}$, the last minimum value, $X(t_m^{\min})$, and the associated time point, $t_m^{\min}$, the next extreme is calculated as:

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

$$X(t_{M+1}^{\max}) = X(t_M^{\max}), \tag{21}$$

with the associated time point being calculated as:

$$t_{M+1}^{\max} = t_m^{\min} + (t_m^{\min} - t_M^{\max}). \tag{22}$$

Symmetric Anchor – In Figure 6, an implementation of the Symmetric Anchor technique (`edge_effect="symmetric_anchor"`) maximum is shown with an orange dot. In Zhao & Huang (2001) and Zeng & He (2004), the technique creates an extreme at the endpoint – this is why this technique is referred to in this paper as the Symmetric Anchor technique. In this paper, this technique has been generalised to conditionally create an extreme depending on the difference in vertical displacement between the last two extrema and the difference in vertical displacement between the last extremum and the end of the signal – this can be referred to as the conditional symmetric anchor technique. The conditional symmetric anchor is calculated as follows – if $\beta L \geq (1-\alpha)L$ where β is the ratio of the vertical height between the last extremum and the end of the time series to the vertical height between the last two extrema, L is the vertical height between the last two extrema, and α (`sym_alpha=0.1`) is the significance level input, then:

$$X(t_{M+1}^{\max}) = X(t_N), \tag{23}$$

with the associated time point being calculated as:

$$t_{M+1}^{\max} = t_N. \tag{24}$$

The above is the method followed (without conditions) in Zhao & Huang (2001) and Zeng & He (2004). If, however, $\beta L < (1-\alpha)L$, then:

$$X(t_{M+1}^{\max}) = X(t_M^{\max}), \tag{25}$$

with the associated time point being calculated as:

$$t_{M+1}^{\max} = t_N + (t_N - t_M^{\max}). \tag{26}$$

The conditional symmetric anchor technique collapses to the symmetric anchor technique when $\alpha = 1$. The other extreme where $\alpha = -\infty$ leads to the following method.

Symmetric – The symmetric technique (`edge_effect="symmetric"`) does not anchor the extrema envelope to the ends of the signal under any condition and is equivalent to the conditional symmetric anchor technique where $\alpha = -\infty$. The values are calculated as follows:

$$X(t_{M+1}^{\max}) = X(t_M^{\max}), \tag{27}$$

with the associated time point being calculated as:

$$t_{M+1}^{\max} = t_N + (t_N - t_M^{\max}). \tag{28}$$

This point is denoted with a grey dot in Figure 6.

Anti-Symmetric – The anti-symmetric (`edge_effect="anti-symmetric"`) approach reflects the univariate signal about both axes – the approximated maximum will be the reflected minimum. It is formally calculated as:

$$X(t_{M+1}^{\max}) = X(t_N) + (X(t_N) - X(t_m^{\min})), \tag{29}$$

with the associated time point being calculated as:

$$t_{M+1}^{\max} = t_N + (t_N - t_m^{\min}). \tag{30}$$

This point is denoted with a green dot in Figure 6. This technique is a more practical variation of that proposed in Zeng & He (2004), where the points are reflected about the axis rather than about the endpoints.

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

Descriptive figures, like Figure 6, detailing the other techniques can be found in "Supplement to: Package **AdvEMDpy**: Algorithmic Variations of Empirical Mode Decomposition in Python". The slope-based family of techniques approach the problem using finite differences by approximating the forthcoming extrema by estimating the forthcoming slopes – the details of which can be found in section 10.3.2 of supplement. Characteristic waves are another family of techniques that approximate the edges using sinusoids with the details in section 10.3.3 of the supplement. Explicit techniques, like the single-neuron neural network (without an activation function) which can be found in section 10.3.4 of the supplement, formally forecast the time series and forecast extrema are taken from the forecast time series.

7.3.2 Recommendations for edge effects in AdvEMDpy package

The edge effect is the most diverse method in this paper with many options available. No method suits every time series. The most robust methods would be the conditional symmetric anchor method or the Coughlin characteristic wave. Both of these methods only use the last two extrema to forecast the next extrema, but the conditional symmetric discard method explicitly takes into account the end of the time series. The conditional symmetric anchor method is therefore recommended. The single neuron neural network has not been as extensively studied and robustified as the other methods, and it is advised to be used with caution.

```
imfs, hts, ifs = \
    emd.empirical_mode_decomposition(knots=knots,
                                     knot_time=knot_time,
                                     edge_effect='symmetric_anchor',
                                     sym_alpha=0.1)[:3]
```

7.4 Detrended fluctuation analysis in AdvEMDpy package

At EMD's core, the algorithm iterates by progressively removing the local mean from the IMF candidate. Owing to the numerous ways of defining and estimating a local mean, local mean estimation should be referred to as detrended fluctuation analysis. Detrended fluctuation analysis was originally introduced in Peng *et al.* (1994) and intended for a different purpose where data were partitioned into equal sets and detrended using discontinuous linear approximations of the local trends. The average of the variances of the detrended data in these partitioned sets was calculated before the subsets were increased in size and the process was repeated. This was done to determine the presence of long-memory processes or rather long-term correlation structures. For our purposes, each set of data (data points between two consecutive knots) is detrended using continuous cubic B-splines to approximate the local trend or local mean. The trends are extracted using local windows defined using the knot sequences. The time series can therefore be estimated as a series of bases and thought of as a sequence of locally defined segments.

In Figure 7, detrended fluctuation analysis is demonstrated using the following time series:

$$f(t) = \cos(2t) + \cos(4t) + \cos(8t) + \epsilon(t), \tag{31}$$

where $\epsilon(t) \in \mathcal{N}(0, 1)$ for $t \in \{t_0, \ldots, t_{1,000}\}$. One can observe in Figure 7 that as the distance between the uniformly placed knots is increased, the fitted splines cannot detect the higher-frequency structures.

In Figure 7, one can observe the relationship between detrended fluctuation analysis and the iterative IMF extraction method known as EMD. In the top image of Figure 7, one can observe the trend as the sum of all the IMFs extracted with a sufficient knot sequence. In the middle image, the trend is approximated with a knot sequence that is insufficient to capture the highest frequency

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

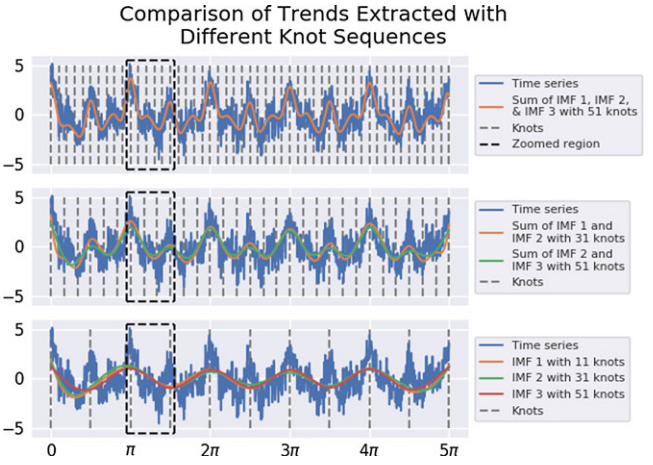

Figure 7. Example time series demonstrating detrended fluctuation analysis of time series with different knot sequences resulting in different trend estimation.

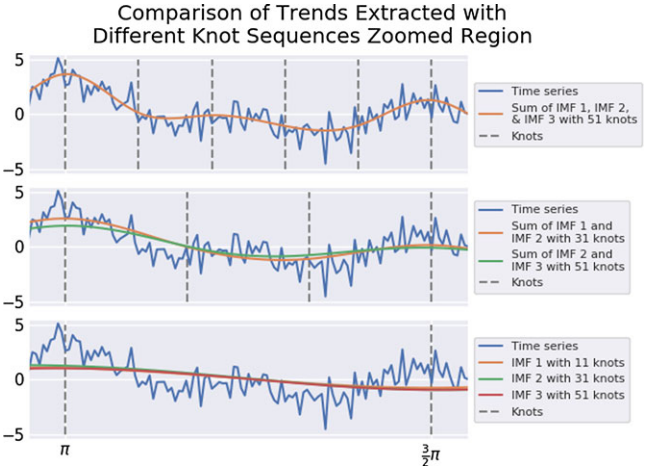

Figure 8. Example time series demonstrating detrended fluctuation analysis of time series with different knot sequences resulting in different trend estimation.

structure. One can also see how closely this trend resembles the sum of IMF 2 and IMF 3 for EMD when the knot sequence is sufficient. Finally, in the bottom image, one can see how the trend is estimated with a wholly insufficient knot sequence compared with the lowest order IMF of each previous sifting procedure.

In Figure 8, the different frequency structures are more easily visible. It can be shown that:

$$\lim_{n\to\infty} \sqrt{\frac{1}{n-1}\sum_{k=1}^{n} \cos^2\left(\frac{2\pi fk}{n}\right)} = \frac{1}{\sqrt{2}} \approx 0.707, \tag{32}$$

with $f \in \mathbb{N}$ and $f << n$. With this in mind, and noting the structure of Equation (31), one can understand the following results:

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

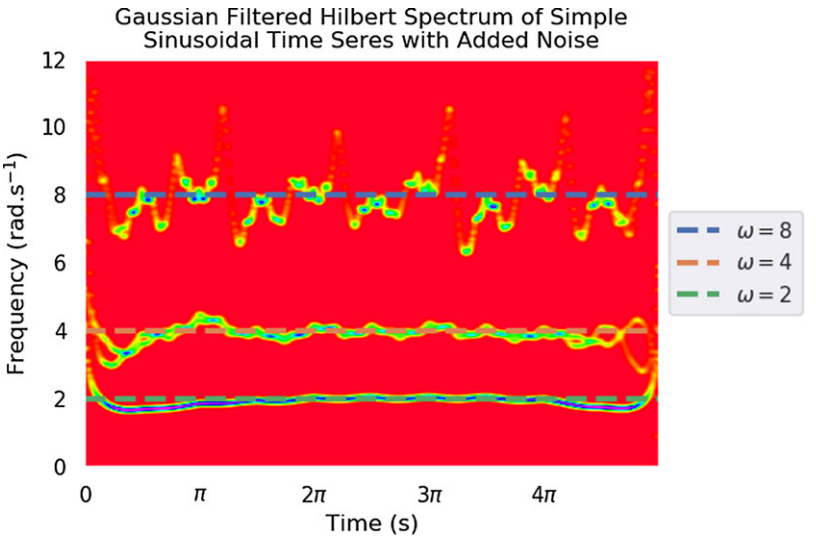

Figure 9. Hilbert spectrum of example time series demonstrating the frequencies of the three IMFs present when sufficient knots are used.

$$\begin{aligned} SD\left(f(t)-\sum_{i=1}^{3} IMF_i^{51}(t)\right) &= 1.005 \approx 1 \\ SD\left(f(t)-\sum_{i=1}^{2} IMF_i^{31}(t)\right) &= 1.558 \\ SD\left(f(t)-IMF_1^{11}(t)\right) &= 2.072, \end{aligned} \tag{33}$$

with IMF_i^j being the ith IMF with knot sequence j. The first equality in Equation (33) most closely calculates the true underlying noise present in the system as a result of the random fluctuations caused by the standard normally distributed Gaussian noise. In the other equalities, the standard deviation calculations are confounded by undetected underlying high-frequency structures with their individual "standard deviations" being approximated by Equation (32). The parallels between detrended fluctuation analysis and EMD should be noted by the user. By noting Figures 7, 8, and 9, one can observe that as a result of the defined knot sequence that fluctuation is measured relative to some frequency band. Each IMF exists in some frequency range, and the fluctuations calculated in Equation (33) measure the local fluctuation relative to some implicit frequency boundary as a result of the knot sequence defined.

In Figure 9, the frequencies of the three constituent structures (apart from the added standard normal Gaussian noise) are visible. The highest frequency structure is perturbed by the noise as expected. EMD can be viewed as a generalisation of detrended fluctuation analysis whereby the trend is decomposed into separate frequency structures in descending order of IF using a defined knot sequence (in general).

The local mean can be estimated using the framework in section 3.3 without any explicit or implicit smoothing. Explicit smoothing that deals simultaneously with smoothing and the edge effects is referred to in the literature as Statistical EMD (SEMD). The base implementation of the detrended fluctuation analysis is as follows:

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

```
imfs, hts, ifs = emd.empirical_mode_decomposition(knots=knots,
                                                  knot_time=knot_time,
                                                  smooth=True,
                                                  smoothing_penalty=0.1,
                                                  dft='envelopes',
                                                  order=15,
                                                  increment=10)[:3].
```

7.4.1 Statistical EMD (P-splines)

This method (`smooth=True`) is introduced in Kim *et al.* (2012). SEMD, in the cubic B-spline envelope setting, can be implemented by introducing a smoothing parameter into the objective function, Equation (17). The specific introduction of a penalty into B-splines is discussed in Eilers & Marx (1996) and is referred to as P-splines. The matricification of the P-spline objective function can be seen below. Second-order smoothing is done on the coefficients, but this can be generalised to higher-order smoothing. The second-order smoothing of the coefficients is incorporated using the $\boldsymbol{D}$ matrix seen below:

$$\boldsymbol{D} = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & -2 & 1 \end{bmatrix}, \tag{34}$$

and with $\boldsymbol{P}$ defined as below,

$$\boldsymbol{P} = \boldsymbol{D}^T\boldsymbol{D}, \tag{35}$$

Equation (17), with $\boldsymbol{s}$, $\boldsymbol{B}$, and $\boldsymbol{c}$ defined as before and with discrete penalty term λ, becomes:

$$DPMSE(\boldsymbol{c}|\boldsymbol{s}) = (\boldsymbol{s} - \boldsymbol{B}\boldsymbol{c})^T(\boldsymbol{s} - \boldsymbol{B}\boldsymbol{c}) + \lambda \boldsymbol{c}^T\boldsymbol{P}\boldsymbol{c}. \tag{36}$$

The magnitude of the penalty term, λ (`smoothing_penalty=0.1`), determines the amount of smoothing. SEMD is implemented separately to the other detrended fluctuation techniques as smoothing can be done irrespective of the detrended fluctuation technique used.

It is highly recommended that `smooth=True` and the smoothing penalty is non-zero when `dft="envelopes"` as the extrema are not guaranteed to satisfy the Schoenberg–Whitney Conditions (SWC) for an arbitrary knot sequence. This will allow envelopes to be fitted even when there are no extrema between successive knot points; otherwise, nonsensical envelopes may result. For an arbitrary set of extrema $\mathbf{y} = \{y_1, y_2, \ldots, y_j\}$ and a cubic B-spline knot sequence $\boldsymbol{\tau} = \{\tau_1, \tau_2, \ldots, \tau_{j+4}\}$, the SWC can be stated as:

$$\tau_i \leq y_i, \leq \tau_{i+1} \; \forall \; i \in \{1, 2, \ldots, j-1\}. \tag{37}$$

In Figure 10, the following time series is plotted to demonstrate the necessity for either the SWC to be satisfied or for the envelopes to be smoothed:

$$g(t) = \cos(t) + \cos(5t), \tag{38}$$

with $t \in [0, 5\pi]$. With $\lambda = 0$ in Equation (36) and with the SWC not being satisfied by either the maxima or the minima, the envelopes are stretched towards zero. This is, unfortunately, unavoidable without either the SWC being satisfied or some form of smoothing.

The other DFA techniques available in this package are demonstrated in Figure 11 and again in "Supplement to: Package **AdvEMDpy**: Algorithmic Variations of Empirical Mode Decomposition

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

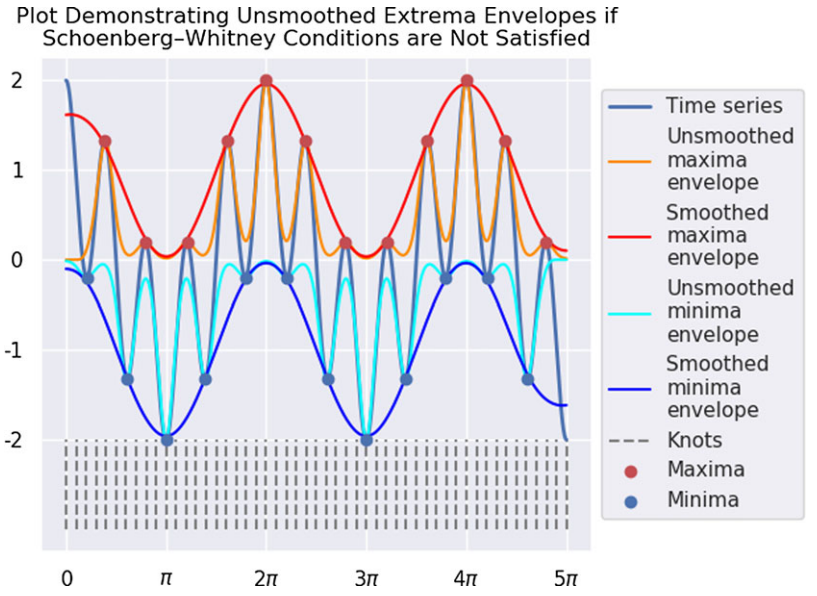

Figure 10. Example time series demonstrating unsmoothed extrema envelopes being fitted when SWC are not satisfied resulting in nonsensical envelopes.

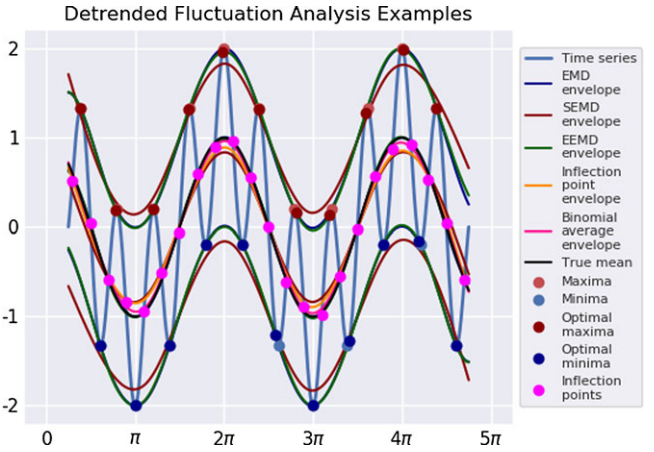

Figure 11. Example time series demonstrating five different local mean estimation techniques through detrended fluctuation analysis.

in Python" for easy reference. The implementation of Enhanced EMD (not called EEMD or E-EMD to avoid confusion with Ensemble EMD (EEMD)) can be found in section 10.4.2 of the supplement – this technique relies on derivatives to optimise, in some sense, the positioning of the "extrema" for the splines. Inflection point interpolation and binomial average interpolation can be found in sections 10.4.3 and 10.4.4, respectively – these two methods are more prone to discontinuities and rely on well-behaved time series, in some sense.

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

7.4.2 Recommendations for detrended fluctuation analysis in AdvEMDpy package

The most studied and utilised local mean estimation technique is the standard envelope technique (`dft="envelopes"`) originally put forward in Huang *et al.* (1998, 1999), and Huang (1999) and, as already mentioned, should be performed with smoothing such that `smooth=True` and `smoothing_penalty=0.1`. This technique is recommended and is intended to be applied after an initial pre-processing or smoothing; otherwise, the first few IMFs may be nonsensical as the noise in the time series will result in the proliferation of extrema and confound the IMFs.

```
imfs, hts, ifs = emd.empirical_mode_decomposition(knots=knots,
                                                  knot_time=knot_time,
                                                  initial_smoothing=True,
                                                  smooth=True,
                                                  smoothing_penalty=0.1,
                                                  dft='envelopes')[:3]
```

7.5 Stopping criteria in AdvEMDpy package

To prevent over-sifting resulting in physically meaningless IMFs and little discernible information about the process under observation, several stopping criteria are provided. The validity of the various stopping criteria warrants further study as some are more related to algorithmic steps than others. The base implementation of the stopping criteria is as follows:

```
imfs, hts, ifs = emd.empirical_mode_decomposition(knots=knots,
                                                  knot_time=knot_time,
                                                  stop_crit='S_stoppage',
                                                  stop_crit_threshold=10,
                                                  mft_theta_1=0.05,
                                                  mft_theta_2=0.5,
                                                  mft_alpha=0.05,
                                                  mean_threshold=10,
                                                  max_internal_iter=30,
                                                  max_imfs=10)[:3].
```

The numerous stopping criteria in this package can be found in the various subsections of section 10.5 in "Supplement to: Package **AdvEMDpy**: Algorithmic Variations of Empirical Mode Decomposition in Python". They are not listed here as they are not the major contributions of this work besides the contribution of section 10.5 being the most complete review of available EMD stopping criteria. A final contribution is the recommendation in the following section which is the most robust stopping criterion available which manages the stopping condition with a logical stopping process (using extrema count variables), as well as managing the total number of external siftings (number of IMFs), the total number of internal siftings, and the modified mean threshold in Modified Condition 2.

7.5.1 Recommendations for stopping criteria in AdvEMDpy package

There are several recommendations for this particular aspect of the algorithm. The most widely used stopping criterion is the Cauchy-type convergence, but this is often unstable and does not converge steadily such as an exponential decaying variable. One should use S Stoppage (`stop_crit="S_stoppage"`) with a threshold of 10 (`stop_crit_threshold=10`). In addition to this criterion, one should impose a value on the modified mean threshold (`mean_threshold=10`), the maximum number of internal iterations

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

(`max_internal_iter=30`), and the maximum allowed number of IMFs (`max_imfs=10`). With all these conditions, the stopping criteria aspect of the algorithm should be sufficiently managed.

```
imfs, hts, ifs = emd.empirical_mode_decomposition(knots=knots,
                                                  knot_time=knot_time,
                                                  stop_crit='S_stoppage',
                                                  stop_crit_threshold=10,
                                                  mean_threshold=10,
                                                  max_internal_iter=30,
                                                  max_imfs=10)[:3]
```

7.6 Spline methods in AdvEMDpy package

All the above techniques are listed with the cubic B-spline implementation of EMD in mind. Other spline techniques are effective when using EMD. The other splines used have a different basis, but because they are both cubic bases, they may be easily mapped onto one another. The base implementation of cubic B-spline EMD is:

```
imfs, hts, ifs = \
    emd.empirical_mode_decomposition(knots=knots, knot_time=knot_time,
                                     spline_method='b_spline')[:3].
```

Cubic Hermite spline interpolation and Akima spline interpolation are also available in this package and are discussed, along with the differences in their respective construction when compared against cubic B-splines, in sections 10.6.1 and 10.6.2, respectively, of "Supplement to: Package **AdvEMDpy**: Algorithmic Variations of Empirical Mode Decomposition in Python".

7.6.1 Recommendations for spline methods in AdvEMDpy package

Cubic Hermite splines and Akima splines lack second-order continuity and as such are not as smooth as cubic B-splines. This may also lead to over-fitting and the higher-frequency IMFs being unwanted smoothed noise. In addition to using the cubic B-spline interpolation method (`spline_method="b_spline"`), the user should also smooth the cubic B-splines (`smooth=True`) for reasons already stated in section 7.4.

```
imfs, hts, ifs = \
    emd.empirical_mode_decomposition(knots=knots,
                                     knot_time=knot_time,
                                     spline_method='b_spline',
                                     smooth=True,
                                     smoothing_penalty=0.1)[:3]
```

7.7 Discrete-time Hilbert transforms in AdvEMDpy package

Despite B-splines having closed-form solutions to the HT, discrete solutions to the HT may be needed for a number of reasons. Here are two widely known and used methods that are included. The base implementation of the DTHT is as follows:

```
imfs, hts, ifs = emd.empirical_mode_decomposition(knots=knots,
                                                  knot_time=knot_time,
                                                  dtht=False,
                                                  dtht_method='fft')[:3].
```

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

The basic DTHT (title from cited literature) and the fast-Fourier transform DTHT (FFT DTHT) can be found in more detail in sections 10.7.1 and 10.7.2, respectively, of "Supplement to: Package **AdvEMDpy**: Algorithmic Variations of Empirical Mode Decomposition in Python". As discussed in the following section, it is often advantageous to compare this output with the closed-form solution and should be output during initial debugging and exploratory investigation of the algorithm.

7.7.1 Recommendations for discrete-time Hilbert transforms in AdvEMDpy package

A DTHT is not output by default, but should the user want a DTHT as well as the closed-form cubic B-spline HT for comparison, the FFT DTHT is recommended. The FFT is significantly faster than the basic DTHT owing to the relationship that exists between the Fourier transform and the HT. Despite this significant increase in computational speed, it is, however, slightly less accurate than the basic DTHT – especially at the edges of the time series.

```
imfs, hts, ifs, _, _, discrete_time_hts, discrete_time_ifs = \
    emd.empirical_mode_decomposition(knots=knots,
                                     knot_time=knot_time,
                                     dtht=True,
                                     dtht_method='fft')
```

7.8 Knot point optimisations in AdvEMDpy package

Two methods are available for the optimisation of knot point allocation. Both methods are in the bisection family of knot optimisation techniques. The first technique simply bisects the domain iteratively until some error bound is met. The next method is a variation on this that extends the domain or diminishes the domain based on some error bound. The base implementation of the knot point optimisation is as follows:

```
imfs, hts, ifs = \
    emd.empirical_mode_decomposition(knots=knots,
                                     knot_time=knot_time,
                                     optimise_knots=0,
                                     knot_method='ser_bisect',
                                     knot_error=10,
                                     knot_lamda=1,
                                     knot_epsilon=0.5)[:3].
```

In section 10.8.1 of "Supplement to: Package **AdvEMDpy**: Algorithmic Variations of Empirical Mode Decomposition in Python", classical bisection is discussed, before the more computationally expensive serial bisection technique is discussed in section 10.8.2 of the supplementary materials. As well as including the formal mathematics, these sections also include descriptive figures to assist with explainability and readability.

7.8.1 Recommendations for knot point optimisations in AdvEMDpy package

There are several options and combinations of options available to the user. Depending on the length and complexity of the time series, it is advised that the algorithm is first run with a uniform set of knots (either user-defined or defaults) with `optimise_knots=0`. Once the algorithm has run without error, the user may opt for the knots to be optimised once at the outset (`optimise_knots=1`). The allowed error (`knot_error=10`) and the allowed minimum

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

knot placement distance (`knot_epsilon=0.5`) are both very time series specific and should be adjusted with care. The Bisection method (`knot_method="bisection"`) is significantly faster than the Serial Bisection method (`knot_method="ser_bisect"`), but does result in slightly more knot points. One should first run the Bisection method with one optimisation at the outset. If one wants knots that dynamically adjust to each potential IMF, then one can use `optimise_knots=2`.

```
imfs, hts, ifs, _, optimised_knots, _, _ = \
    emd.empirical_mode_decomposition(knots=knots,
                                     knot_time=knot_time,
                                     optimise_knots=1,
                                     knot_method='bisection',
                                     knot_error=10,
                                     knot_lamda=1,
                                     knot_epsilon=0.5)
```

7.9 Ensemble empirical mode decomposition in AdvEMDpy package

This is a noise-assisted data analysis (NADA) technique introduced in Wu & Huang (2009) that utilises white noise. This technique relies on the ability of the algorithm to discern consistent structures in the presence of different sets of randomised white noise. The base implementation of the EEMD is implemented as follows:

```
imfs, hts, ifs = emd.empirical_mode_decomposition(knots=knots,
                                                  knot_time=knot_time,
                                                  ensemble=False,
                                                  ensemble_sd=0.5,
                                                  ensemble_iter=10)[:3].
```

The original time series is median filtered to approximate the original level of noise in the system. The median is removed from the time series before the standard deviation is calculated, σ_s. The noise added to the system has a mean of zero and a standard deviation of $\sigma_s \times \sigma_e$, where σ_e (`ensemble_sd=0.5`) is the chosen level of standard deviation in the ensemble system as a fraction of the original standard deviation in the system. The IMFs are calculated for this modified time series and stored before a new set of noise is added to the original time series and the sifting process is repeated – the procedure is repeated a predetermined number of iterations (`ensemble_iter=10`).

Further applications of this technique will benefit from random additions of other colours of noise such as violet noise, blue noise, pink noise, and red noise (also known as Brownian noise), which all have non-constant power spectral densities, compared with white noise that has a constant power spectral density. This technique, with the random addition of other colours of noise, is called full-spectrum ensemble empirical mode decomposition (FSEEMD). This technique is experimental and is also included with the package in the script entitled `emd_experimental.py` for completeness sake.

This method performs a similar task, utilising a different methodology, to ICA-EMD, introduced in van Jaarsveldt *et al.* (2021), which is finding the most consistent structures among noisy data. EEMD proceeds by introducing noise to the time series and isolating IMFs from the noisy time series. This task is repeated several times and IMFs are simply averaged. A more sophisticated sorting technique such as the minimum description length principle (introduced in Fayyad & Irani (1993)) should be used to ensure structures of similar frequency are grouped as in van Jaarsveldt *et al.* (2021). ICA-EMD proceeds by applying independent component analysis (ICA)

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

directly to all the noisy IMFs to isolate the most consistent structure, before EMD is performed on the ICA component to isolate IMFs.

7.9.1 Recommendations for ensemble empirical mode decomposition in AdvEMDpy package

This technique shows promise, but it should be implemented with care. The implementation of the EEMD method removes some additional features such as the output of coefficients, knots, as well as the DTHT and IF from the corresponding DTHT. One should apply this technique after applying EMD for comparison.

```
imfs, hts, ifs = emd.empirical_mode_decomposition(knots=knots,
                                                  knot_time=knot_time,
                                                  ensemble=True,
                                                  ensemble_sd=0.5,
                                                  ensemble_iter=10)
```

8. Illustrative examples of EMD method

We provide three examples, two of which are included in an online Supplementary Appendix entitled "Supplement to: Package **AdvEMDpy**: Algorithmic Variations of Empirical Mode Decomposition in Python" accompanying this manuscript. The intentions are firstly to demonstrate the application of the toolbox in a real data example and secondly to compare the accuracy of the methods implemented in the proposed toolbox versus other existing EMD packages, i.e. comparing **AdvEMDpy** against PyEMD 0.2.10 and emd 0.3.3.

The Supplementary materials contain two additional worked examples (one synthetic and one real-world) to illustrate the increased accuracy of **AdvEMDpy** when compared against PyEMD 0.2.10 and emd 0.3.3 in other applications of relevance to the actuarial audience. In section S.1 of the supplementary materials, **AdvEMDpy** is shown to more accurately (MSE) resolve the driving function of the Duffing Equation, whereas in section S.2 of the supplementary materials, **AdvEMDpy** is shown to more accurately (MSE) resolve the annual Carbon cycle in the atmosphere.

8.1 United Kingdom of Great Britain and Northern Ireland birth rates

Trends in birth rates occur for a variety of reasons such as global events (World War 2), national governmental intervention (such as the NHS providing the contraceptive pill and the Abortion Act), and societal trends (such as people delaying childbearing years to later ages). These structures can be seen in the plotted time series shown in Figure 12 along with numerous other structures. An annual cycle is faintly visible in Figure 12 with the annual structure representing the first IMF in the EMD of the data in Figure 12 and available here: Mortality Database (2022).

The IF of the first IMF can be seen in Figure 13. Little analysis can be performed on this IF as noise is still present. The highest intensities can be seen every 9 months at a few points in the previous century. It is, however, clear that some structure is present. By using EMD-X11, first proposed in van Jaarsveldt *et al.* (2021), one can improve the resolution of the IF – this can be seen in Figure 14.

Better analysis can be performed on Figure 14. After World War 2 and before the implementation of the Abortion Act (1968) and the NHS providing of contraceptive pills to unmarried women, there is a clear increasing in frequency and intensity structure. After the contraceptive pill became available to the general public, this structure began to depreciate until it finally stabilised at the beginning of the 1990s. After this period, a clear annual structure is observed with only a

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

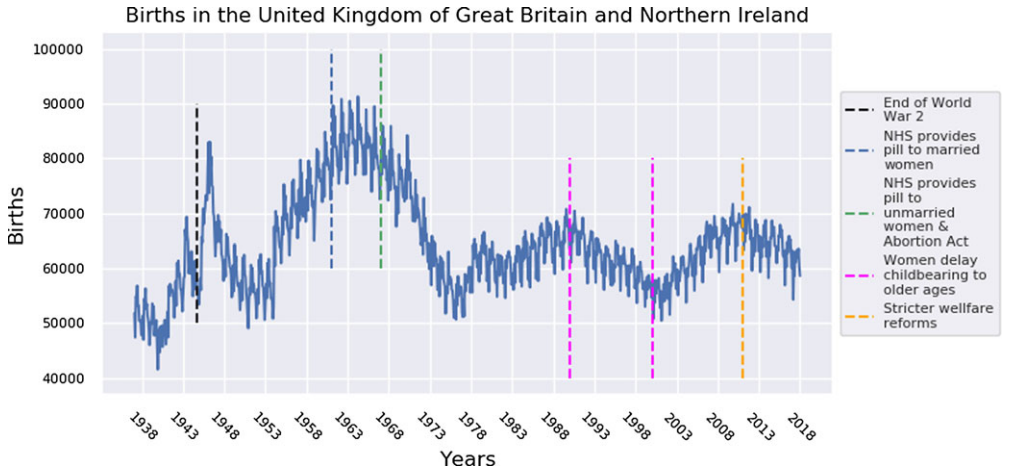

Figure 12. Monthly births in the United Kingdom from January 1938 until December 2018 inclusive, with some notable features and causality.

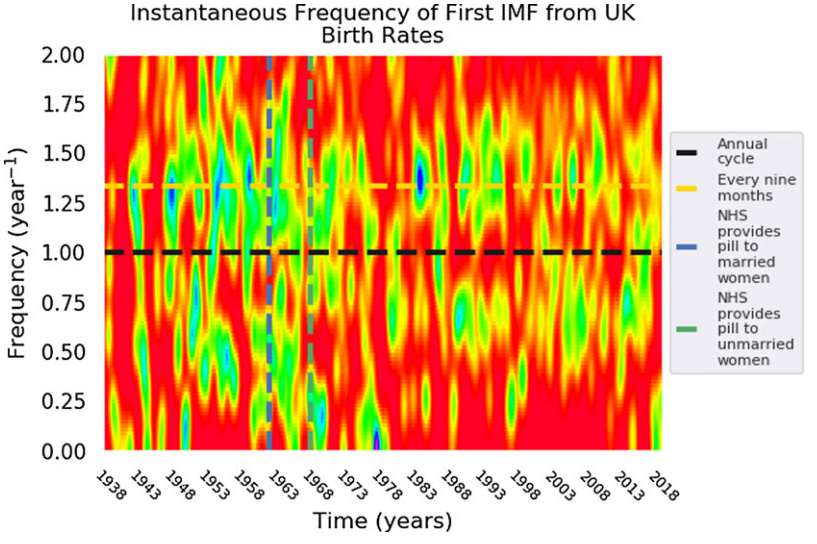

Figure 13. IF of first IMF extracted using EMD of monthly births in the United Kingdom from January 1938 until December 2018 inclusive, with some features and limits included.

minor break in the 2000s during the most recent surge in birth rates. The increased resolution of EMD-X11 has allowed improved analysis of the birth rates.

8.2 United Kingdom of Great Britain and Northern Ireland mortality trends

In Figures 15 and 16, the total number of deaths in the United Kingdom can be seen for women and men, respectively. One of the most prevalent structures in both figures is the exponential decrease in the infant (aged 0–5 non-inclusive) mortality rate over the past century. This was

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

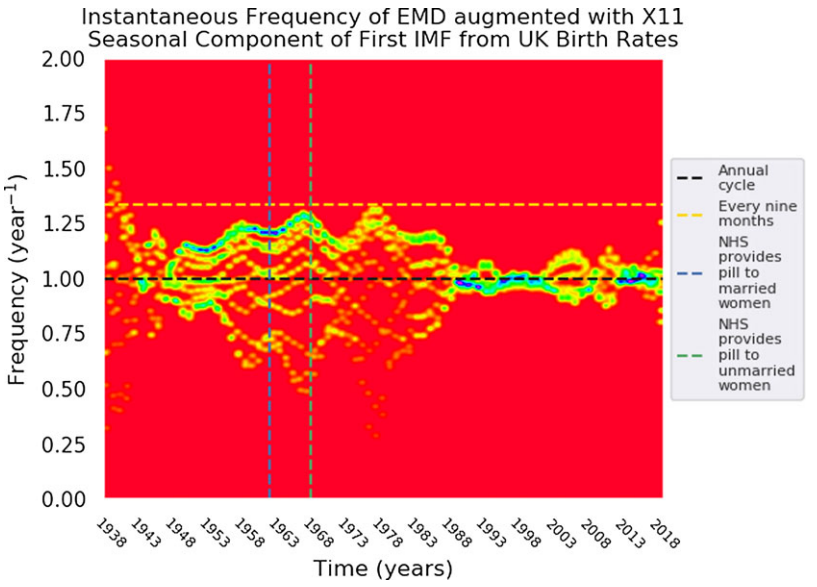

Figure 14. IF of first IMF extracted using EMD and refined using X11 of monthly births in the United Kingdom from January 1938 until December 2018 inclusive, with some features and limits included.

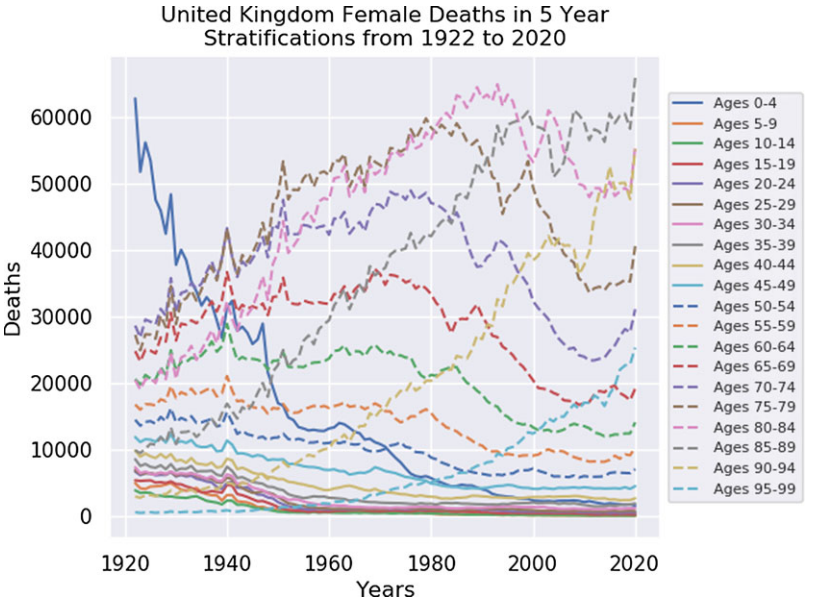

Figure 15. Annual deaths among females in the United Kingdom from 1922 until 2020 inclusive, in a 5-year stratification.

part of a major campaign by many European countries and others worldwide to decrease infant mortality. Another, less obvious, almost meta-structure can be seen as the increasing of year and deaths in the maxima as a result of the significant increase in births after World War 1. This meta-structure would be more difficult to analyse with most modern techniques, but EMD is able to iteratively decompose the structures and associate these maxima.

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

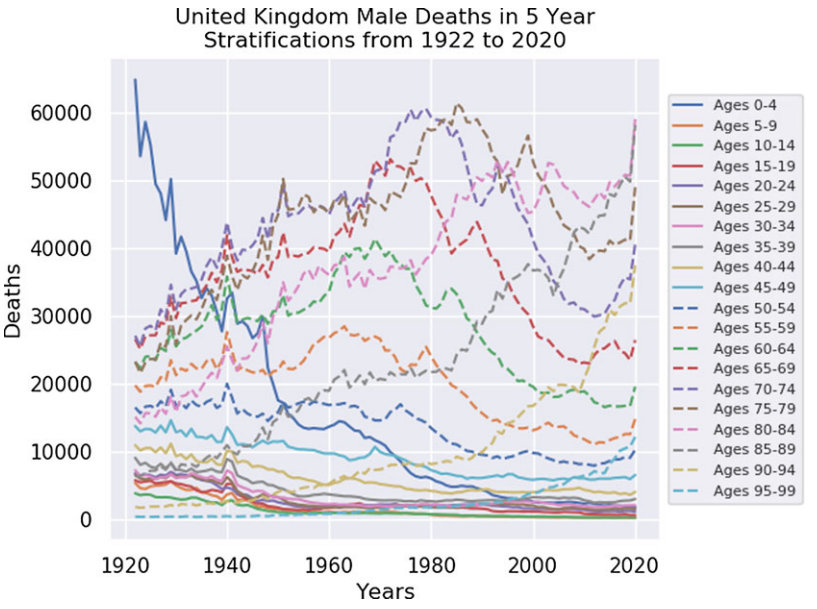

Figure 16. Annual deaths among males in the United Kingdom from 1922 until 2020 inclusive, in a 5-year stratification.

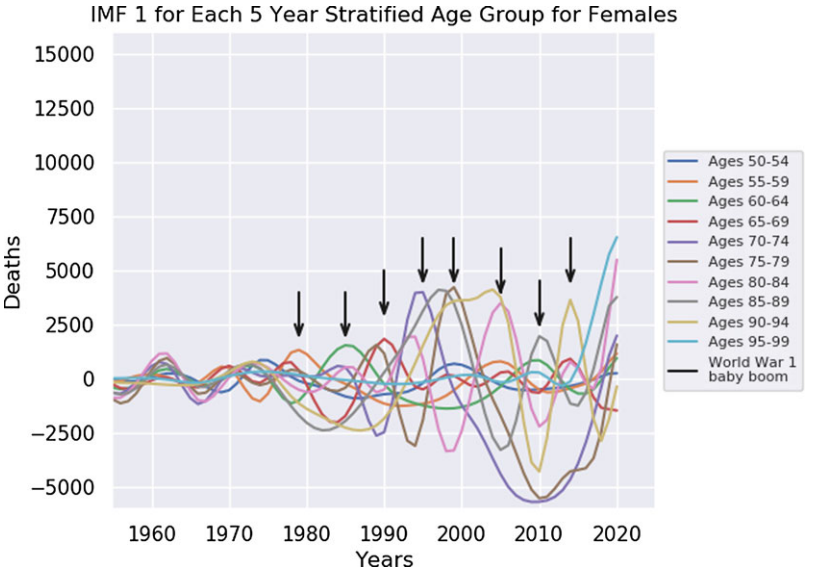

Figure 17. IMF 1 of deaths among females in the United Kingdom from 1922 until 2020 inclusive, in a 5-year stratification.

In Figures 17, 18, 19, and 20, the first IMF for each of the five year stratifications over the age of 50 are plotted. In Figure 17, the cyclical nature of this mortality data is clear, but the true underlying driving-force behind this structure is obscured. By plotting the deaths based on when people were born, as in Figure 18, the fewer people dying who were born between 1915–1920 is as a result of the World War 1 killing a statistically significant number of people in the general population.

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

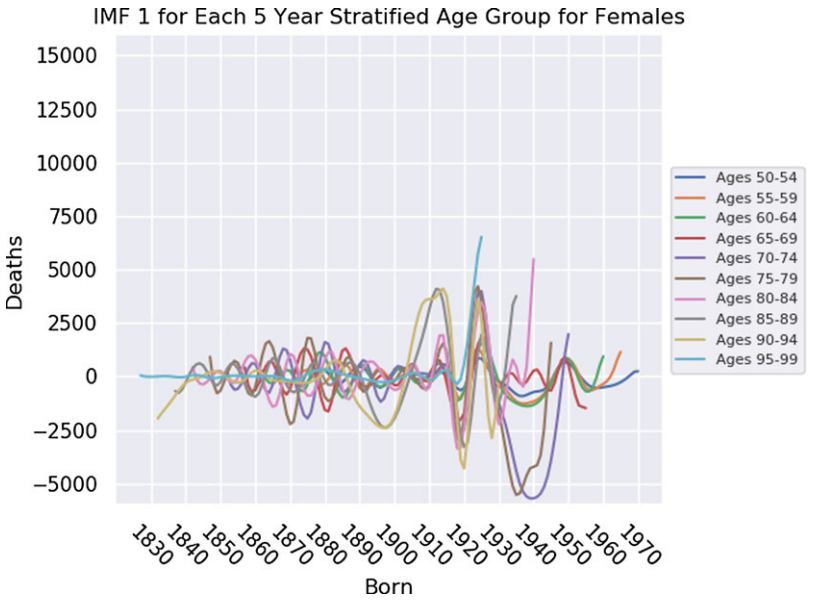

Figure 18. IMF 1 of deaths among females in the United Kingdom from 1922 until 2020 inclusive, in a 5-year stratification, sorted by birth year.

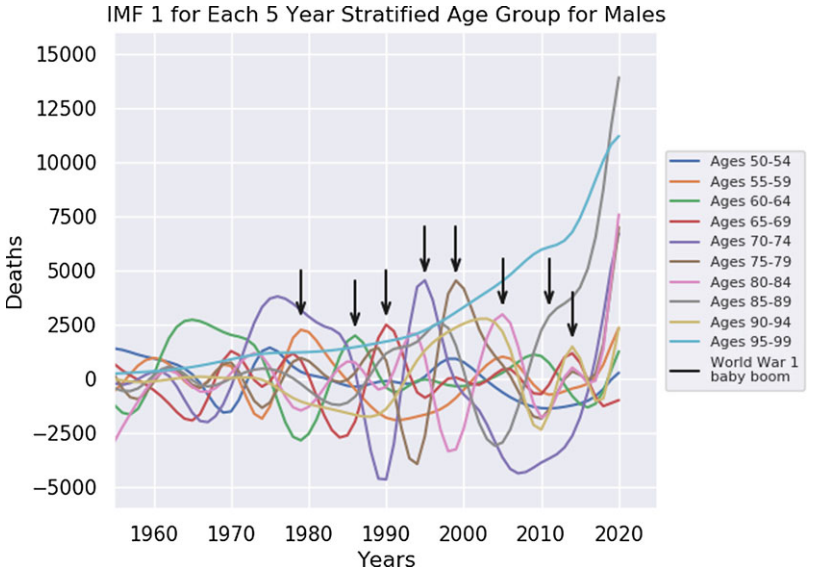

Figure 19. IMF 1 of deaths among males in the United Kingdom from 1922 until 2020 inclusive, in a 5-year stratification.

This is followed by the post-World War 1 baby boom in the 1920s. The analogous data for men in the United Kingdom are plotted in Figures 19 and 20 – the same structures are observable in these figures as can be expected, but the amplitude of the World War 1 event structure among males is more pronounced owing to the statistically more men that died during World War 1.

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

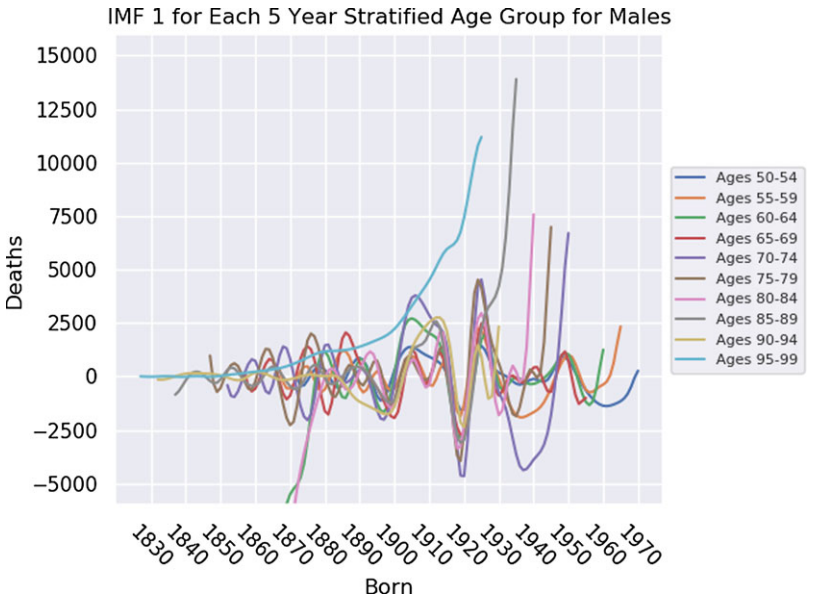

Figure 20. IMF 1 of deaths among males in the United Kingdom from 1922 until 2020 inclusive, in a 5-year stratification, sorted by birth.

9. Conclusion

AdvEMDpy seeks to provide customisability not present in other available EMD packages both within Python and across other programming languages such as R and MATLAB. Pre-processing of time series is sorely neglected in other packages, which is addressed in section 7.2 with future developments in compressive sampling EMD (CSEMD) falling in this section and addressed in `emd_experimental.py` and `experimental.ipynb`. The edge effect, which is the most ubiquitous problem in all of time series analysis, is also given a rather scant treatment in other packages. Four separate families of techniques are reviewed thoroughly in section 7.3 with many others to follow. The approximation of the local mean structure through detrended fluctuation analysis is also given a very brief treatment in other packages. This paper and package seek to formalise the treatment and introduce several modern viable alternatives such as inflection point interpolation and binomial average interpolation which are presented in section 7.4.

Several stopping criteria and spline choices are provided with most of these also being available in other packages. Stopping criteria and splines are addressed in sections 7.5 and 7.6, respectively. Most recent developments in EMD have been focused on edge effects and detrended fluctuation analysis. Stopping criteria all try to prevent over-sifting, whereas spline choices attempt to arrive at a desirable trade-off between accuracy and over-fitting. Knot optimisation has not been addressed in other packages and provides valuable new insight into spline choices and over-fitting and is addressed in section 7.8. In addition to the stable viable options available in the package, FSEEMD (an extension of EEMD discussed in section 7.9) and CSEMD are briefly discussed with demonstrative examples being shown in `emd_experimental.py` and `experimental.ipynb`.

This package is intended for both new users of EMD and advanced users who have experience with these decomposition and analysis algorithms. Several simple scripts are included in this paper, as well as in `aas_figures_replication_script.py` and the Jupyter Notebooks included with this package. In both sections S.1 and S.2 of the supplementary materials in "Supplement to: Package **AdvEMDpy**: Algorithmic Variations of Empirical Mode Decomposition in Python", **AdvEMDpy** is shown to exceed PyEMD 0.2.10 and emd 0.3.3. In section S.1,

https://doi.org/10.1017/S1748499523000088 Published online by Cambridge University Press

AdvEMDpy is shown to more accurately resolve the underlying driving function that drives the dynamic equilibrium of the system. Finally, in section S.2, **AdvEMDpy** is shown to significantly exceed the other methods in resolving the IF of the annual underlying cycle. This package will benefit many users, both those new to EMD and those experienced in the interrelated fields of EMD, time series analysis, detrended fluctuation analysis, and time series filtering and graduation.

In section 8, the birth and mortality data of the United Kingdom are analysed using this package. Some clear features are present in Figure 12, section 8.1. The World War 2 baby boom is apparent with a sharp decrease before increasing to a maximum in the 1960s before stabilising as a result of the increased availability of birth control. Figure 14 demonstrates the increased availability of birth control by a stabilising annual cycle. The HHT has allowed increased resolution of United Kingdom birth data. In section 8.2, the United Kingdom death rates are analysed using vanilla EMD and used to accurately demonstrate potentially catastrophic (for insurers) shifts in death rates as a result of past events such World War 1 and the resulting baby boom (Figures 18 and 20) – EMD has proven to be an invaluable technique in any actuarial analyst's handbook. Please see "Supplement to: Package **AdvEMDpy**: Algorithmic Variations of Empirical Mode Decomposition in Python" for further examples as well as the public GitHub repository for numerous other useful examples and experimental techniques.

Supplementary material. To view supplementary material for this article, please visit http://doi.org/10.1017/S1748499523000088.

References

Ahmad, **S.**, **Agrawal**, **S.**, **Joshi**, **S.**, **Taran**, **S.**, **Bajaj**, **V.**, **Demir**, **F.** & **Sengur**, **A.** (2020). Environmental sound classification using optimum allocation sampling based empirical mode decomposition. *Physica A: Statistical Mechanics and its Applications*, **537**(1–11), 122613. https://doi.org/10.1016/j.physa.2019.122613

Aziz, **S.**, **Naqvi**, **S.**, **Khan**, **M.** & **Aslam**, **T.** (2020). Electricity theft detection using empirical mode decomposition and K-nearest neighbors. In *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)* (pp. 1–5). IEEE. https://doi.org/10.1109/ICETST49965.2020.9080727

Bowman, **A.** & **Evers**, **L.** (2017). Nonparametric smoothing lecture notes, technical report, The University of Glasgow, Glasgow. https://warwick.ac.uk/fac/sci/statistics/apts/students/resources-1314/ns_notes.pdf

Campi, **M.**, **Peters**, **G.**, **Azzaoui**, **N.** & **Matsui**, **T.** (2021). Machine learning mitigants for speech based cyber risk. *IEEE Access*. https://doi.org/10.1109/ACCESS.2021.3117080

Chen, **Q.**, **Huang**, **N.**, **Riemenschneider**, **S.** & **Xu**, **Y.** (2006). A B-spline approach for empirical mode decompositions. *Advances in Computational Mathematics*, **24**(1–4), 171–195. https://doi.org/10.1007/s10444-004-7614-3

Chiew, **F.**, **Peel**, **M.**, **Amirthanathan**, **G.** & **Pegram**, **G.** (2005). Identification of oscillations in historical global streamflow data using empirical mode decomposition. In S. Franks, T. Wagener, E. Bøgh, H. Gupta, L. Bastidas, C. Nobre & C. Galvão (Eds.), *Regional Hydrological Impacts of Climatic Change - Hydroclimatic Variabiltiy (Brazil)* (pp. 53–62), Vol. 296. International Association of Hydrological Sciences. https://www.cabdirect.org/cabdirect/abstract/20053083220

Coughlin, **K.** & **Tung**, **K.** (2004). 11-Year solar cycle in the stratosphere extracted by the empirical mode decomposition method. *Advances in Space Research*, **34**(2), 323–329. https://doi.org/10.1016/j.asr.2003.02.045

Craven, **P.** & **Wahba**, **G.** (1978). Smoothing noisy data with spline functions. *Numerische Mathematik*, **31**(4), 377–403. https://doi.org/10.1007/BF01404567

de Boor, **C.** (1978). *A Practical Guide to Splines*. *Applied Mathematical Sciences*, Vol. **27**. Springer-Verlag, New York, USA. https://doi.org/10.2307/2006241

de Freitas Lucena, **R.**, **Costa**, **R.**, **Castelar**, **I.** & **de Lima**, **F.** (2021). Dynamic analysis of criminal behavior: an application of empirical mode decomposition. *International Journal of Economics and Finance*, **13**(4), 1–47.

Deering, **R.** & **Kaiser**, **J.** (2005). The use of a masking signal to improve empirical mode decomposition. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP05)* (pp. 485–488), Vol. 4. IEEE. https://doi.org/10.1109/ICASSP.2005.1416051

Drakakis, **K.** (2008). Empirical mode decomposition of financial data. *In International Mathematical Forum* (pp. 1191–1202), Vol. **3**. Citeseer.

Du, **X.**, **Li**, **Z.**, **Bi**, **F.**, **Zhang**, **J.**, **Wang**, **X.** & **Shao**, **K.** (2012). Source separation of diesel engine vibration based on the empirical mode decomposition and independent component analysis. *Chinese Journal of Mechanical Engineering*, **25**(3), 557–563. https://doi.org/10.3901/CJME.2012.03.557

Egambaram, **A.**, **Badruddin**, **N.**, **Asirvadam**, **V.** & **Begum**, **T.** (2016). Comparison of envelope interpolation techniques in Empirical Mode Decomposition (EMD) for eyeblink artifact removal from EEG. In *IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES)* (pp. 590–595). IEEE. https://doi.org/10.1109/IECBES.2016.7843518

Eilers, **P.** & **Marx**, **B.** (1996). Flexible smoothing with B-splines and penalties. *Statistical Science*, **11**(2), 89–121. https://doi.org/10.1214/ss/1038425655

El-Kafrawy, **N.**, **Hegazy**, **D.** & **Tolba**, **M.** (2014). Features extraction and classification of EEG signals using empirical mode decomposition and support vector machine. In A. Hassanien, M. Tolba & A. Taher Azar (Eds.), *Advanced Machine Learning Technologies and Applications* (pp. 189–198). Springer. https://doi.org/10.1007/978-3-319-13461-1_19

Fayyad, **U.** & **Irani**, **K.** (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)* (pp. 1022–1027), Vol. 2. http://hdl.handle.net/2014/35171

Guhathakurta, **K.**, **Mukherjee**, **I.** & **Chowdhury**, **A.** (2008). Empirical mode decomposition analysis of two different financial time series and their comparison. *Chaos, Solitons & Fractals*, **37**(4), 1214–1227. https://doi.org/10.1016/j.chaos.2006.10.065

Hao, **H.**, **Yu**, **F.** & **Q. Li**. (2020). Soil temperature prediction using convolutional neural network based on ensemble empirical Mode decomposition. *IEEE Access*, **9**, 4084–4096. https://doi.org/10.1109/ACCESS.2020.3048028

Huang, **N.** (1999). Computer Implemented Empirical Mode Decomposition Method, Apparatus and Article of Manufacture. Patent. US Patent 5,983,162.

Huang, **N.**, **Hu**, **K.**, **Yang**, **A.**, **Chang**, **H.**, **Jia**, **D.**, **Liang**, **W.**, **Yeh**, **J.**, **Kao**, **C.**, **Juan**, **C.**, **Peng**, **C.**, **Meijer**, **J.**, **Wang**, **Y.**, **Long**, **S.** & **Wu**, **Z.** (2016). On Holo-Hilbert spectral analysis: a full informational spectral representation for nonlinear and non-stationary data. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **374**(2065), 1–21. https://doi.org/10.1098/rsta.2015.0206

Huang, **N.**, **Shen**, **Z.** & **Long**, **S.** (1999). A new view of nonlinear water waves: the Hilbert spectrum. *Annual Review of Fluid Mechanics*, **31**(1), 417–457. https://doi.org/10.1146/annurev.fluid.31.1.417

Huang, **N.**, **Shen**, **Z.**, **Long**, **S.**, **Wu**, **M.**, **Shih**, **H.**, **Zheng**, **Q.**, **Yen**, **N.**, **Tung**, **C.** & **Liu**, **H.** (1998). The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, **454**(1971), 903–995. https://doi.org/10.1098/rspa.1998.0193

Huang, **N.** & **Wu**, **Z.** (2008). A review on Hilbert-Huang transform: method and its applications to geophysical studies. *Reviews of Geophysics*, **46**(2), (RG2006) 1–23. https://doi.org/10.1029/2007RG000228

Huang, **Z.** & **Ling**, **B.** (2022). Sleeping stage classification based on joint quaternion valued singular spectrum analysis and ensemble empirical mode decomposition. *Biomedical Signal Processing and Control*, **71**(A), 103086 (1–14). https://doi.org/10.1016/j.bspc.2021.103086

Jin, **X.**, **Yang**, **N.**, **Wang**, **X.**, **Bai**, **Y.**, **Su**, **T.** & **Kong**, **J.** (2020). Hybrid deep learning predictor for smart agriculture sensing based on empirical mode decomposition and gated recurrent unit group model. *Sensors*, **20**(5), 1334 (1–20). https://doi.org/10.3390/s20051334

Kim, **D.**, **Kim**, **K.** & **Oh**, **H.** (2012). Extending the scope of empirical mode decomposition by smoothing. *EURASIP Journal on Advances in Signal Processing*, **168**(2012), 1–17. https://doi.org/10.1186/1687-6180-2012-168

Kim, **D.** & **Oh**, **H.** (2018). *EMD: Empirical Mode Decomposition and Hilbert Spectral Analysis*. https://CRAN.R-project.org/package=EMD R package version 1.5.8.

Laszuk, **D.** (2020). *EMD-signal*: *Implementation of the Empirical Mode Decomposition (EMD) and its variations*. https://pypi.org/project/EMD-signal/ Python package version 0.2.10.

Lin, **C.**, **Chiu**, **S.** & **Lin**, **T.** (2012). Empirical mode decomposition-based least squares support vector regression for foreign exchange rate forecasting. *Economic Modelling*, **29**(6), 2583–2590. https://doi.org/10.1016/j.econmod.2012.07.018

Liu, **Z.**, **Huang**, **F.** & **Li**, **B.** (2015). Analysis on characteristics and influential factors of grain yield fluctuation in China based on empirical mode decomposition. *Transactions of the Chinese Society of Agricultural Engineering*, **31**(2), 7–13.

MathWorks. (2021). *MATLAB*. The Mathworks, Inc., Natick, Massachusetts, USA. https://www.mathworks.com/ MATLAB Version R2021a.

Human Mortality Database. (2022). *United Kingdom Total Population*. https://www.mortality.org/Country/Country?cntr=GBR_NP

Nava, **N.**, **Di Matteo**, **T.** & **Aste**, **T.** (2018). Financial time series forecasting using empirical mode decomposition and support vector regression. *Risks*, **6**(1), 7 (1–21). https://doi.org/10.3390/risks6010007

Ortigueira, **M.** (2021). *Empirical Mode Decomposition*. https://www.mathworks.com/matlabcentral/fileexchange/21409-empirical-mode-decomposition MATLAB package version 1.0.0.0.

Peng, **C.**, **Buldyrev**, **S.**, **Havlin**, **S.**, **Simons**, **M.**, **Stanley**, **H.** & **Goldberger**, **A.** (1994). Mosaic organization of DNA nucleotides. *Physical Review E: Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, **49**(2), 1685–1689. https://doi.org/10.1103/PhysRevE.49.1685

Python Core Team. (2019). *Python: A Dynamic, Open Source Programming Language*. *Python Software Foundation*, Amsterdam, Netherlands. https://www.python.org/ Python Version 3.7.4.

Quinn, **A.** (2020). *emd: Empirical Mode Decomposition*. https://pypi.org/project/emd/ Python package version 0.3.2.

Rezaei, **D.** & **Taheri**, **F.** (2010). Health monitoring of pipeline Girth Weld using empirical mode decomposition. *Smart Materials and Structures*, **19**(5), 055016 (1–18). https://doi.org/10.1088/0964-1726/19/5/055016

Rezaei, **D.** & **Taheri**, **F.** (2011). Damage identification in beams using empirical mode decomposition. *Structural Health Monitoring*, **10**(3), 261–274.

Rilling, **G.**, **Flandrin**, **P.** & **Goncalves**, **P.** (2003). On empirical mode decomposition and its algorithms. In *IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing* (pp. 8–11), Vol. **3**. NSIP-03, Grado (I). https://hal.inria.fr/inria-00570628

R Core Team. (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

Salisbury, **J.** & **Sun**, **Y.** (2004). Assessment of chaotic parameters in nonstationary electrocardiograms by use of empirical mode decomposition. *Annals of Biomedical Engineering*, **32**(10), 1348–1354. https://doi.org/10.1114/B:ABME.0000042223.87320.de

Sun, **X.**, **Yang**, **P.**, **Li**, **Y.**, **Gao**, **Z.** & **Zhang**, **Y.** (2012). Robust heart beat detection from photoplethysmography interlaced with motion artifacts based on empirical mode decomposition. In *Proceedings of 2012 IEEE-EMBS International Conference on Biomedical and Health Informatics* (pp. 775–778). IEEE. https://doi.org/10.1109/BHI.2012.6211698

Tabrizi, **A.**, **Garibaldi**, **L.**, **Fasana**, **A.** & **Marchesiello**, **S.** (2014). Influence of stopping criterion for sifting process of Empirical Mode Decomposition (EMD) on roller bearing fault diagnosis. In G. Dalpiaz, R. Rubini, G. D'Elia, M. Cocconcelli, F. Chaari, R. Zimroz, W. Bartelmus & M. Haddar (Eds.), *Advances in Condition Monitoring of Machinery in Non-Stationary Operations* (pp. 389–398). Springer-Verlag, Berlin, Heidelberg. 9783642393488 https://doi.org/10.1007/978-3-642-39348-8_33

Torres, **M.**, **Colominas**, **M.**, **Schlotthauer**, **G.** & **Flandrin**, **P.** (2011). A complete ensemble empirical mode decomposition with adaptive noise. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4144–4147). IEEE. https://doi.org/10.1109/ICASSP.2011.5947265

van Jaarsveldt, **C.**, **Peters**, **G.**, **Ames**, **M.** & **Chantler**, **M.** (2021). Tutorial on Empirical Mode Decomposition: Basis Decomposition and Frequency Adaptive Graduation in Non-Stationary Time Series. Available at SSRN 3913330. https://doi.org/10.2139/ssrn.3913330

Wahba, **G.** (1990). *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 6th edition. Philadelphia, Pennsylvania, USA. Based on a series of 10 lectures at Ohio State University at Columbus, Mar. 23-27, 1987.

Wang, **J.** & **Wang**, **J.** (2017). Forecasting stochastic neural network based on financial empirical mode decomposition. *Neural Networks*, **90**, 8–20. https://doi.org/10.1016/j.neunet.2017.03.004

Wu, **C.**, **Huang**, **J.**, **Minasny**, **B.** & **Zhu**, **H.** (2017). Two-dimensional empirical mode decomposition of heavy metal spatial variation in agricultural soils, Southeast China. *Environmental Science and Pollution Research*, **24**(9), 8302–8314. https://doi.org/10.1007/s11356-017-8511-x

Wu, **F.** & **Qu**, **L.** (2008). An improved method for restraining the end effect in empirical mode decomposition and its applications to the fault diagnosis of large rotating machinery. *Journal of Sound and Vibration*, **314**(3–5), 586–602. https://doi.org/10.1016/j.jsv.2008.01.020

Wu, **Z.** & **Huang**, **N.** (2009). Ensemble empirical mode decomposition: a noise-assisted data analysis method. *Advances in Adaptive Data Analysis*, **1**(1), 1–41. https://doi.org/10.1142/S1793536909000047

Zeng, **K.** & **He**, **M.** (2004). A simple boundary process technique for empirical mode decomposition. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)* (pp. 4258–4261), Vol. 6. IEEE. https://doi.org/10.1109/IGARSS.2004.1370076

Zhao, **J.** & **Huang**, **D.** (2001). Mirror extending and circular spline function for empirical mode decomposition method. *Journal of Zhejiang University A: Science*, **2**(3), 247–252. https://doi.org/10.1007/BF02839453

Zheng, **Z.**, **Fan**, **J.**, **Liu**, **H.** & **Zeng**, **D.** (2010). The analysis and predictions of agricultural drought trend in Guangdong Province based on empirical mode decomposition. *Journal of Agricultural Science*, **2**(4), 170–179.

Cite this article: van Jaarsveldt C, Ames M, Peters GW and Chantler M (2023). Package AdvEMDpy: Algorithmic variations of empirical mode decomposition in Python, *Annals of Actuarial Science*, **17**, 606–642. https://doi.org/10.1017/S1748499523000088